به نام خدا

دانشگاه کردستان

گروه کامپیوتر

درس آزمایشگاه سیستم‌های عامل

تهیه:

صادق سلیمانی

<ویرایش سوم >

زمستان ۸۹

فهرست

پیشگفتار

سیستم‌های عامل از پیچیده‌ترین و بزرگترین نرم‌افزارها در حیطه‌ی مهندسی نرم‌افزار به شمار می‌آیند که در مقیاس وسیع و توسط کاربران با سطح دانش‌های متفاوت مورد استفاده قرار می‌گیرند. نظر به چنین اهمیتی، درسی سه واحدی برای آشنایی با مفاهیم سیستم‌های عامل در دوره‌ی کارشناسی مهندسی و علوم کامپیوتر و فناوری اطلاعات در نظر گرفته شده است. آشنایی ملموس و عمیق‌تر با مفاهیم نظری مطرح شده در سیستم‌های عامل جز با یک دوره‌ی عملی در این زمینه امکان‌پذیر نمی‌باشد.

تجربه‌ی بیش از شش سال تدریس در رابطه با سیستم‌های عامل و فعالیت حرفه‌ایی در زمینه‌ی سیستم‌عامل و شبکه و تألیف کتاب "افزایش کارایی رایانه‌های شخصی" و فقدان مرجع متناسب در زمینه‌ی آزمایشگاه سیستم‌های عامل، بنده را بر آن داشت که با توجه به موارد زیر، در این زمینه مستندی را به خصوص برای دانشگاه‌های فاقد آزمایشگاه مجهز مجزا تهیه نمایم.

در دوره‌ی کارشناسی کامپیوتر، درس آزمایشگاه سیستم‌های عامل با محدودیت‌های مهمی روبرو است که باید در تدوین مستندی برای آن مد نظر قرار گیرد تا بهره‌وری هرچه بیشتری برای دانشجویان به همراه داشته باشد. به عنوان مثال زمان ارائه برای چنین درسی، یک واحد عملی که در بیشتر دانشگاه‌ها معادل با دو ساعت است در نظر گرفته شده، که مجال عرضه‌ی بسیاری مفاهیم به صورت پیوسته را محدود می‌نماید. همچنین اشتراک کامپیوترهای مورد استفاده برای آزمایشگاه سیستم‌های عامل با سایر دروس در بسیاری از دانشگاه‌ها، ممکن است امکان تغییرات مهم و سیستمی را از دانشجویان سلب نماید.

سیستم‌های عامل مختلفی در دنیای امروز مورد استفاده هستند:

- **Windows** که محصول شرکت مایکروسافت است و خود دارای ویرایش‌های متعددی برای کاربران خانگی و سازمان‌ها و شرکت‌های بزرگ است، مانند: Windows XP، Windows Vista، Windows Server 2003 و.... این سیستم‌عامل گزینه‌های جدید و پیشرفته‌ی سیستم‌عامل‌های امروزی از قبیل Multiuser، MultiTasking، Virtual Memory و ... را داراست.
- **UNIX** که برای اولین بار در سال ۱۹۶۸ درآزمایشگاه AT&T Bell عرضه شد نیز خود دارای ویرایش‌های عمده‌ایی برای کار درمقیاس خانگی (Linux) و صنعتی است. همچنین این

سیستم‌عامل نیز گزینه‌های جدید و پیشرفته‌ی سیستم‌عامل‌های امروزی از قبیل Multiuser، MultiTasking، Virtual Memory و ... را داراست.

- MAC Operating System که در سال ۱۹۸۴ در شرکت Apple و برای کامپیوترهای شخصی Macintosh عرضه شد و تاکنون نسخه‌های مختلفی از آن عرضه شده است و برای اغلب کاربران ایرانی ناآشناست.
- Novell Netware که در سال ۱۹۸۴ با انگیزه تسهیل اشتراک فایل و دیسک در بستر شبکه عرضه شد، محصولی از شرکت Novell با پشتیبانی IBM است که طرفداران زیادی حتی در کشور ایران پیدا کرد ولی اکنون به تدریج در حال برچیده شدن است.
- نسخه‌های مختلف سیستم‌عامل‌های دیگر مانند Solaris که مبتنی بر UNIX است و سایر سیستم‌عامل‌های خاص منظوره برای استفاده‌های مختلف مانند دستگاه‌های تلفن همراه، انواع دستگاه‌ها و لوازم صنعتی و خانگی برای استفاده‌های معمول یا بلادرنگ[1] و ...

با این حال بر کسی پوشیده نیست که به دلیل شرایط خاص حق کپی[2] در ایران، پرطرفدارترین و پرکاربردترین سیستم‌عامل مورد استفاده، نسخه‌های مختلف سیستم‌عامل ویندوز محصول شرکت مایکروسافت است. جهت تسهیل و تسریع آزمایش‌ها، این نکته نیز در تألیف این مستند مورد نظر بوده است. این تألیف با در نظر گرفتن محدودیت‌های ذکر شده تدوین شده است و موارد زیر از ویژگی‌های آن به شمار می‌آیند:

- امکان ارائه‌ی اغلب آزمایش‌ها در جلسات نزدیک به دو تا سه ساعته
- امکان ارائه‌ی درس آزمایشگاه سیستم‌های عامل در سایت‌های کامپیوتر مشترک با سایر دروس
- تکیه بر سیستم‌عامل‌های رایج در محیط دانشگاه و کار، برای آزمایش‌ها
- تکیه بر نکات دوسویه دارای جنبه‌های علمی و فنی، به امید استفاده‌ی سودمند آن برای دانشجویان در هر دو محیط علم و کار
- اشاره به مفاهیم مفید دیگر مرتبط با سیستم‌عامل مانند BIOS، نرم‌افزارهای سودمند و ...
- تکیه بر تمرین‌های اضافی مفید و مکمل

در صورت عدم وجود سیستم‌عامل نصب شده و آماده برای استفاده در آزمایشگاه، استفاده از نرم‌افزارهای ماشین مجازی مانند Microsoft Virtual PC یا Sun Virtual Box پیشنهاد می‌شود. همانگونه که ما نیز

[1] RealTime
[2] Copy Right

در این درس از Sun Virtual Box برای اجرا و استفاده‌ی سیستم عامل لینوکس ویرایش Ubuntu جهت آزمایش‌های مرتبط در دانشگاه کردستان استفاده کردیم.

لازم به ذکر است که در این ویرایش تعداد ده آزمایش تدوین شده که برای یک ترم درس کفایت نمی‌کند و لازم است دست کم چند آزمایش دیگر بر آن افزوده شود تا هم مدرس در انتخاب و اولویت‌بندی آزمایش‌ها، آزادی عمل بیشتری داشته باشد و هم مباحث مهم بیشتری پوشش داده شود. موارد زیر از اهداف آتی این مستند به حساب می‌آید:

- گنجاندن جلساتی در رابطه با سایر سیستم‌عامل‌های غیر از ویندوز و لینوکس
- گنجاندن آزمایش‌های مربوط به برنامه‌نویسی سیستم در ویندوز و مرتبط با مفاهیم برنامه‌نویسی چند نخی، پردازش رویداد در ویندوز و ابزارهای همزمانی در سیستم عامل (راهنما[1] و ناظر[2]) و .... تا در صورت عدم پرداخت به آن‌ها در سایر دروس از جمله برنامه‌نویسی پیشرفته، در این درس پوشش داده شود.
- افزودن Socket Programming، تا در صورت صلاحدید مدرس و عدم طرح این مسأله در سایر دروس از قبیل شبکه یا مهندسی فناوری اطلاعات، در این آزمایشگاه مطرح شود.
- به هنگام‌سازی آزمایش‌های سیستم‌عامل ویندوز، منطبق برویرایش‌های جدید و قابلیت‌های جدید مطرح شده.

۱-۱- نسخه الکترونیکی کتاب و راهنمای مربی

بی‌شک مطالب عرضه شده خالی از اشکال نیست و مطالب محدودی از سیستم عامل را در اختیار دانشجویان راغب قرار می‌دهد. از این رو، تلاش مستمری برای تکمیل و غنی‌تر شدن آن در جریان است و در این مسیر مؤلف، پیشاپیش از نظرات و پیشنهادات ارسالی شما به آدرس ایمیل info@ITVirtualLab.com استقبال می‌نماید. خواهشمند است ایمیل‌های خود را تحت عنوان OS-Lab Manual ارسال نمایید.

همچنین نسخه‌ی راهنمای مربی برای اساتید محترمی که قصد استفاده از این مستند را به عنوان مرجع یا کمک درسی در نظر دارند، آماده است و ایشان در صورت تمایل می‌توانند در ایمیلی با عنوان OS-Lab Manual Guide، راهنمای مذکور را دریافت دارند.

[1] Semaphore
[2] Monitor

در صورت نیاز آزمایش‌ها به استفاده از نرم‌افزار یا فایلی خاص، آن فایل یا نرم‌افزار در صفحه‌ی درس، در وب‌سایت www.ITVirtualLab.com در دسترس قرار خواهد گرفت.

۱-۲- برنامه‌درسی

برای برگزار شدن هرچه بهتر این درس و دستیابی به اهداف آموزشی مورد نظر آن، پیشنهادهای زیر قابل لحاظ نمودن است. لازم به ذکر است پیشنهادهای مذکور بر اساس تجربه عملی برگزاری درس و گرفتن بازخوردهای مثبت، تدوین شده است:

- آزمایش‌ها در گروه‌های حداکثر سه نفره و حداقل دو نفره باید انجام شود و گروه یک نفره قابل قبول نیست.
- در صورت عدم وجود تمهیدات مناسب در سایت، دانشجو موظف است پس از هر آزمایش در پایان ساعت، تغییرات سیستم را به حالت اول برگرداند.
- پیش از حضور در هرجلسه باید پیش‌آگاهی خوانده شود و از آن کوییز گرفته خواهد شد.
- به ازای هر جلسه، باید گزارش کار برای آزمایش‌هایی که دارای گزارش‌کار هستند پر شود و تحویل داده شود.
- پرسش‌های تکمیلی با رعایت مهلت تعیین شده، برای تکمیل درک آزمایش‌های انجام پذیرفته، بایستی تحت عنوان تکلیف، تحویل داده شود.
- امکان تعریف یک پروژه‌ی مکمل نیز همراه با این درس وجود دارد، که به خاطر محدود بودن تعداد آزمایش‌های این ویرایش از مستند، می‌توان انواع مسایل برنامه‌نویسی مرتبط با هماهنگی فرایندها، شناخت سیستم‌عامل‌های جدید یا خاص‌منظوره و ... را به عنوان موضوع عرضه نمود.
- جلسه‌ی آزمون نهایی نظری و عملی، آخرین جلسه‌ی کلاس‌ها طبق اعلام آموزش دانشگاه است. جلسه‌ی آزمون عملی در همان ساعت کلاس و آزمون نظری به مدت نیم ساعت و به صورت هماهنگ برای تمام گروه‌هاست.
- حداکثر تعداد غیبت‌های موجه ۲ جلسه است و بیش از آن به صورت خودکار دانشجو حذف خواهد شد. جلسه‌ی پیش از آزمون نهایی، جلسه‌ی جبرانی است و دانشجویانی که یک یا دو غیبت مجاز داشته‌اند موظف‌اند، آزمایش‌های انجام نشده را در این جلسه انجام داده و گزارش‌کار مربوطه را تحویل دهند.

- به خاطر اهمیت تکالیف در فهم دروس پیشین و پسین، اکیداً توصیه می‌شود، تکالیف در زمان مقرر تحویل داده شود.

۱-۳- سپاسگزاری

لازم می‌دانم بدینوسیله از خانم‌ها آرزوایزدنیا و نیلوفرخانقاهی، دانشجویان مقطع کارشناسی دانشگاه آزاد اسلامی واحد دماوند، خانم طاهره احمدیان، آقایان یاسر قادری، ضیاءالدین نجفیان، مازیار منوچهری و مجتبی ملک‌پور دانشجوی کارشناسی مهندسی فناوری اطلاعات دانشگاه کردستان و آقای محمد سعید مؤذن دانشجوی کارشناسی مهندسی فناوری اطلاعات دانشگاه تبریز که در تهیه‌ی پیشنویس‌های اولیه این مستند و ویرایش آن همکاری نمودند تشکر نمایم. همچنین تهیه‌ی این اثر را به جناب آقای دکتر محسن صدیقی مشکنانی (دانشگاه صنعتی شریف، پردیس کیش) به خاطر این که ایده‌ی آن را از دوران تحصیل کارشناسی اینجانب، در ذهن بنده ایجاد نمودند، مدیونم. از دکتر علی بهلولی زفره (دانشگاه اصفهان) به خاطر القای الگوی مفید نحوه‌ی عرضه‌ی دروس آزمایشگاه نیز سپاسگزارم. همچنین از سایر دانشجویان درس آزمایشگاه سیستم‌های عامل در دانشگاه‌های صنعتی اصفهان، علم و صنعت بهشهر، دانشگاه آزاد اسلامی دماوند و دانشگاه کردستان به خاطر تحقیقات و نظرات جهت‌دارشان در ارائه‌ی مفیدتر درس متشکرم.

صادق سلیمانی
info@ITVirtualLab.com
گروه مهندسی کامپیوتر و فناوری اطلاعات دانشکده مهندسی
دانشگاه کردستان
بهمن‌ماه ۱۳۸۹

آزمایش اول

تنظیم کارایی سیستم‌عامل

۲-۱- مقدمه

در این آزمایش که مقدمه‌ای خواهد بود بر قابلیت‌های سیستم‌عامل در جهت تنظیم کارایی، به پراهمیت‌ترین پارامترهایی که در افزایش کارایی سیستم‌عامل نقش دارند پرداخته خواهد شد. آزمایش تدوین شده تحت سیستم‌عامل ویندوز است و برای ویرایش‌های مختلف آن قابل اعمال است، اما برای استفاده بر ویندوز XP تنظیم شده است.

۲-۲- هدف

آشنایی با مهمترین پارامترهای مرتبط با کارایی سیستم‌عامل ویندوز و نحوه‌ی تنظیم آن‌ها

۲-۳- پیش‌آگاهی

کامپیوتر نیز مانند هر وسیله‌ی دیگری در اثر استفاده‌ی مکرر دچار کاهش کارایی خواهد شد و همانند هر وسیله‌ی دیگری نیاز به تنظیم و نگهداری درست دارد. از مشکلات اساسی کاربران در کار با کامپیوترهای شخصی و سازمان در کار با کارگزارها[1]، کاهش کارایی سیستم پس از مدتی استفاده و نصب و حذف برنامه‌های کاربردی و انتقال و جابه‌جایی داده است. تا حدی که گاهی کاربران دارای سیستم‌های با سخت‌افزار قوی، سرعت و کارایی سیستم خود را از سرعت سیستم ضعیف‌تر اما جدید، پایین‌تر احساس می‌کنند.

راهکارهای افزایش کارایی سیستم‌های کامپیوتری در دو رده‌ی عمده جای می‌گیرند:

۱. سخت‌افزاری
۲. نرم‌افزاری (تنظیم سیستم‌عامل)

جهت افزایش کارایی سیستم از طریق سخت‌افزاری باید به گلوگاه‌های مهم کارایی آن توجه نمود که معمولاً در این میان پردازنده[2] و حافظه‌ی اصلی[3] از اهمیت بیشتری برخوردارند و باید آن‌ها را ارتقا داد.

اما راهکار صحیح که هدف اصلی این آزمایش است، تنظیم و بررسی نرم‌افزاری (سیستم‌عامل) در آغاز است، تا پس از آن بتوان در رابطه با نیاز به ارتقای سخت‌افزار نیز اظهار نظر نمود.

تنظیمات نرم‌افزاری گزینه‌های متعددی را در برمی‌گیرد که در این آزمایش، به ترتیب اولویت، به برخی از مهمترین و مؤثرترین آن‌ها پرداخته خواهد شد.

۲-۳-۱- حذف عناصر غیرضروری از Startup

Startup مکانی در سیستم‌عامل ویندوز است که چند میانبر[4] از برنامه‌های کابردی مختلف در آن قرار دارد و ویندوز پس از بارگذاری سیستم‌عامل، تمام میانبرهای موجود در آن را بررسی نموده و برنامه‌ی متناظر را به حافظه‌ی اصلی بارگذاری می‌کند.

اکثر برنامه‌های کاربردی در زمان نصب، میانبری از خود در آنجا قرار می‌دهند. دلیل این امر، دسترسی سریع به برنامه، توسط کاربر، در زمان اجرای آن است، نیز بسیاری از چنین برنامه‌هایی از طریق اینترنت و بدون اطلاع کاربر، آخرین به‌هنگام‌سازی‌ها و اصلاحات خود را در صورت اتصال به اینترنت، دانلود

[1] Servers
[2] CPU
[3] RAM
[4] Shortcut

می‌کنند. البته برخی از چنین برنامه‌هایی برای ادامه فعالیت صحیح سیستم ضروری هستند و باید همیشه در Startup قرار داشته باشند، به عنوان مثال ویروس‌کش‌ها را می‌توان نام برد.

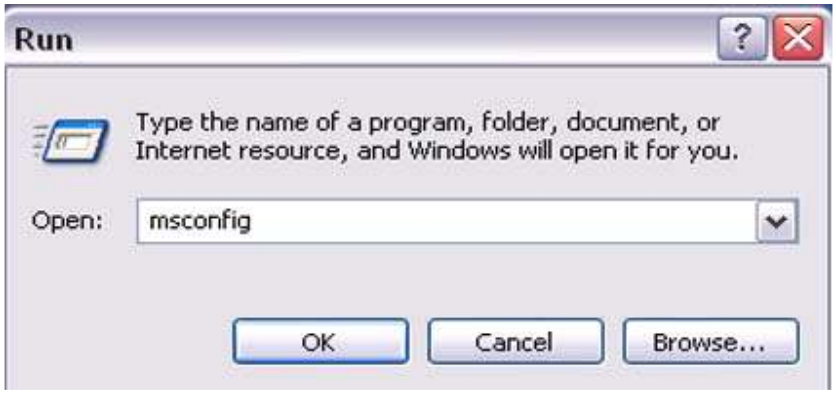
بارگذاری چنین برنامه‌هایی که معمولاً حجیم هم هستند، به دو صورت کارایی سیستم را کاهش می‌دهد:

۱. افزایش زمان شروع به کار ویندوز
۲. کاهش میزان فضای حافظه‌ی اصلی در دسترس برای اجرای برنامه‌ها

لازم به ذکر است که برخی برنامه‌ها را نیز کاربر خود در Startup قرار می‌دهد، زیرا به دلایلی ممکن است مایل باشد که همیشه در شروع به کار سیستم اجرا شوند، به عنوان مثال می‌توان برنامه‌های فرهنگ لغت را نام برد. همچنین Startup مکانی مستعد برای استقرار بسیاری از ویروس‌های کامپیوتری است تا در شروع به کار سیستم، بلافاصله خود را در حافظه‌ی اصلی مستقر نمایند. بیشتر برنامه‌هایی که از طریق Startup در حافظه‌ی مجازی بارگذاری می‌شوند، در سینی سیستم[1] نیز نمایش داده می‌شوند. نمایی از سینی سیستم در زیر نمایش داده شده است:

شکل ۲-۱ نمای System Tray

دسترسی به عناصر Statrtup از طریق نوشتن فرمان msconfig در Start->Run است.



شکل ۲-۲ دسترسی به عناصر Startup از طریق Run

۲-۳-۲- تنظیم حافظه‌ی مجازی

حجم واقعی برخی برنامه‌ها مانند Adobe Photoshop، Visual Studio .NET یا ... بسیار بزرگتر از حافظه‌ی اصلی است، اما در اجرای آن‌ها هیچ خللی ایجاد نمی‌شود و به سادگی قابل استفاده‌اند! واقعیت آن

[1] System Tray

است که سیستم‌عامل، تمام برنامه را به یک‌باره از حافظه‌ی جانبی (Hard Disk) به حافظه‌ی اصلی وارد نمی‌کند، بلکه همواره بخشی از آن را در حافظه‌ی اصلی نگهداری می‌کند که اکنون برای اجرا نیاز است. مکانی از حافظه‌ی جانبی (دیسک سخت) که برنامه‌ها پیش از بارگذاری در حافظه‌ی اصلی برای اجرا، به آن منتقل می‌شوند و تنها جزئی از برنامه به حافظه‌ی اصلی منتقل می‌شود که اکنون برای اجرا ضروریست، حافظه‌ی مجازی[1] نامیده می‌شود. بقیه‌ی قسمت‌های برنامه تا پایان اجرا در حافظه‌ی مجازی می‌مانند تا در زمان نیاز به حافظه‌ی اصلی منتقل شوند. حافظه‌ی مجازی در تمام سیستم‌عامل‌های مدرن امروزی وجود دارد و نام‌های دیگری نیز برای آن استفاده می‌شود، از قبیل: Swap File یا Page File.

تنظیم درست حافظه‌ی مجازی، نقش مهمی در کارایی و سرعت سیستم، به خصوص برای برنامه‌های سنگین، ایفا می‌کند. کم بودن حجم آن سبب عدم امکان بارگذاری برنامه‌های حجیم به حافظه‌ی اصلی می‌شود و زیاد بودن اندازه‌ی آن سبب هدر رفتن فضای دیسک سخت می‌شود.

سیستم عامل ویندوز، خود حافظه‌ی مجازی را تنظیم می کند اما این تنظیم بر اساس پایین‌ترین نیاز کاربر است و به تناسب افزایش درخواست کاربر و برنامه‌های کاربردی، آن را افزایش می‌دهد. به بیان دیگر، در سیستم عامل ویندوز دو مقدار برای حافظه‌ی مجازی در نظر گرفته شده است: مقدار کمینه[2] و بیشینه[3]. در سیستم عامل لینوکس نیز در همان ابتدای نصب سیستم‌عامل باید بخشی را به عنوان حافظه‌ی مجازی (Swap File) مجزا نمود. لازم به ذکر است که خود سیستم‌عامل نیز به عنوان یک برنامه‌ی بزرگ، از حافظه مجازی استفاده می‌کند.

در ادامه نکات مهم تنظیم حافظه‌ی مجازی به صورت خلاصه ذکر خواهند شد:

۱. توصیه می‌شود حافظه‌ی مجازی بر درایوی مجزا از درایو سیستم‌عامل نصب شود، زیرا تعدد ارجاعات و ترافیک درخواست برای مکان‌های مختلف آن درایو ایجاد می‌شود و عملیات مدیریت فایل با تأخیر همراه خواهد شد و در نتیجه کارایی کاهش خواهد یافت.
۲. توصیه می‌شود اندازه‌ی حافظه‌ی مجازی دو و نیم تا سه برابر حافظه‌ی اصلی کنونی مورد استفاده باشد.
۳. توصیه می‌شود پیش از تنظیم حافظه‌ی مجازی در یک درایو، درایو مورد نظر را Defragment نماییم، تا فضای حافظه‌ی مجازی یک قسمتی و پیوسته باشد.

[1] Virtual Memory
[2] Minimum
[3] Maximum

۴. توصیه می‌شود مقدار بیشینه و کمینه برای حافظه‌ی مجازی، هر دو همان مقدار بیشینه باشند. زیرا در غیر این‌صورت سیستم‌عامل همیشه با مقدار کمینه برای حافظه مجازی شروع خواهد کرد و در صورت نیاز به توسعه‌ی آن، ممکن است حافظه‌ی مجازی چند قطعه شود و متعاقباً ارجاعات مکرر به نقاط مختلف دیسک سبب کاهش کارایی گردد.
۵. پیشنهاد می‌شود در صورت امکان، به خصوص برای ویندوزهای کارگزار، یک درایو مجزا برای حافظه‌ی مجازی اختصاص داده شود.

۲-۳-۲-۱- مزایای حافظه‌ی مجازی

اولین مزیت حافظه‌ی مجازی آن است که اجرای برنامه‌ها با اندازه‌ی فوق‌العاده بزرگ را امکان‌پذیر می‌سازد، زیرا برای اجرای یک برنامه، نیاز نیست که همه‌ی آن را به حافظه‌ی اصلی بارگذاری نمود. مزیت دوم در افزایش درجه‌ی چندبرنامگی است، بدین ترتیب که می‌توان از هر برنامه، تنها یک جزء کوچک از آن را که اکنون برای اجرا نیاز است، به حافظه بارگذاری نمود و در واقع حافظه‌ی اصلی را برای اجرای برنامه‌های بیشتر، خالی نمود. البته ملاحظات متعددی در این رابطه باید لحاظ شود که تفصیل آن را می‌توان در کتاب‌های سیستم‌عامل مطالعه نمود.

۲-۳-۳- استفاده از نرم‌افزارهای[1] سودمند همراه با سیستم‌عامل

منظور از نرم‌افزارهای سودمند، نرم‌افزارهای کاربردی کوچکی هستند که توسط تولیدکننده‌ی سیستم‌عامل و همراه با سیستم‌عامل به صورت رایگان عرضه می‌شوند. به عنوان مثال Windows Media Player یا Paint نرم‌افزارهای رایگانی هستند که همراه با سیستم‌عامل ویندوز عرضه می‌شوند. چنین نرم‌افزارهایی می‌توانند سیستمی نیز باشند. اهمیت نرم‌افزارهای سودمند سیستمی در آن است که تولیدکننده‌ی یک سیستم‌عامل تجاری مانند ویندوز، بهتر از هر تولیدکننده‌ی نرم‌افزار دیگری از ویژگی‌ها و خصوصیات داخلی آن آگاه است. تا جاییکه بسیاری از مؤلفین در زمینه‌ی تنظیم و افزایش کارایی رایانه‌های شخصی، استفاده از یک نرم‌افزار ساده مانند Scan Disk را در بسیاری از موارد توصیه می‌کنند.

سه مورد از نرم‌افزارهای سودمند ویندوز که در ادامه بررسی خواهند شد عبارتند از:

۱. Scan Disk
۲. Disk Defragment
۳. Disk Cleanup

[1] Utilities

Scan Disk -۱-۳-۳-۲

این نرم‌افزار که در ویرایش‌های Windows 98 و ماقبل، Scan Disk نام داشت، اکنون به عنوان گزینه‌ی Error Checking نام برده می‌شود. شاید این گزینه به صورت مستقیم به افزایش سرعت سیستم مرتبط نباشد، اما نقش مهمی در رفع ایرادهای متعدد مرتبط با کاهش کارایی دارد. به عنوان مثال، یکی از مؤثرترین کاربردهای Scan Disk، یافتن ناسازگاری داده، پس از خاموش شدن ناگهانی سیستم است که در اولین شروع مجدد کار سیستم، یک پنجره‌ی آبی ظاهر خواهد شدکه از کاربر خواهد خواست تا برای رفع ناسازگاری احتمالی داده، حافظه‌ی جانبی را چک نماید. این نرم‌افزار سودمند با آدرس زیر قابل دسترسی است:

Right click on one drive → properties → tools tab → error-checking

Disk Defragment -۲-۳-۳-۲

زمانیکه کاربر برنامه‌های کاربردی‌ایی را بر روی حافظه‌ی ثانویه (دیسک سخت) خود نصب می‌کند، ویندوز به دنبال قطاع‌های[1] مجاوری بر روی دیسک می‌گردد، تا برنامه‌ی کاربردی را در آنجا قرار دهد. اگر همه‌ی قسمت‌های یک پرونده مجاور هم قرار گیرند، دیسک سخت می‌تواند با کارایی بیشتری به پرونده دسترسی پیدا کند. اما اغلب اوقات ساختار پرونده در خلال نصب قطعه قطعه[2] می‌شود و قسمت‌هایی از برنامه کاربردی (که ممکن است در مکان‌های پیوسته ممکن جا نشوند) در نقاط تصادفی سراسر دیسک جای می‌گیرند. این امر هد خواندن دیسک سخت را وادار به گذر از هر جای دیسک برای دستیابی به داده مورد نیاز می‌کند. این امر برای برخی کارها سبب صرف زمان بیشتری خواهد شد. Microsoft Defrag داده‌های پراکنده‌ی دیسک سخت را مجددا به‌صورت پیوسته در می‌آورد. و همچنین پرونده‌های[3] برنامه‌هایی را که شما اغلب استفاده می‌کنید، به خاطر دسترسی سریعتر، در آغاز[4] دیسک قرار می‌دهد.

این ابزار در ارتباط مستقیم با مدیریت حافظه‌ی ثانویه است که از فصول آخر کتاب‌های درسی سیستم‌عامل به حساب می‌آید. وظیفه‌ی آن قرار دادن اجزای پراکنده‌ی فایل‌ها به صورت پیوسته است که با این کار دو مزیت حاصل می‌شود، اول اینکه زمان دسترسی را تسریع می‌کند و دوم اینکه از تجمیع فضاهای خالی پراکنده نیز فضای خالی یک‌دست ایجاد می‌شود. این گزینه، به‌خصوص در کامپیوترهای کارگزار که ارجاعات مکرر به حافظه‌ی جانبی دارند، سبب افزایش سرعت خواهد شد.

نکاتی چند در رابطه با استفاده از Disk Defragment لازم به بیان است:

---

[1] Sectors
[2] Fragmented
[3] File
[4] Front

- بازه‌ی زمانی انجام این برنامه‌ی سودمند به میزان فعالیت و استفاده‌ی از سیستم بستگی دارد و در سیستم‌های فعال، انجام هر دو هفته یک‌بار آن سودمند خواهد بود.
- از پیش‌نیازهای Defrag کردن این است که ۱۵٪ از ظرفیت درایو مورد نظر خالی باشد.
- توصیه می‌شود در زمان استفاده از Disk Defragment از سایر برنامه‌های ویندوز استفاده نشود، نیز به دلیل وقت‌گیر بودن این فرایند، پیشنهاد می‌گردد که در زمان‌هایی که سیستم برای مدتی غیرفعال است، آن را مورد استفاده قرار داد.

برای دسترسی به این ابزار، مسیر زیر را طی نمایید:

Start -> Programs -> Accessories -> System Tools -> Disk Defragmenter

۲-۳-۳-۳- Disk Cleanup

به صورت پیش‌فرض، سیستم عامل ویندوز موظف است در زمان اتمام (بسته شدن) یک برنامه، تمام فایل‌های ناشی از نیاز یا محاسبات موقت آن را از پوشه‌ایی که در درایو ویندوز و در پوشه‌ی Windows قرار دارد و نام آن Temp است پاک نماید. همچنین فایل‌های بدون استفاده و موقت ممکن است سبب ناسازگاری‌هایی در اجرای سایر برنامه‌ها گردند. علاوه بر آن امکان حذف فایل‌های باقی‌مانده ناشی از نصب سایر برنامه‌ها و محتوی Recycle Bin نیز توسط این نرم‌افزار سودمند قابل پاکسازی است.

برای دسترسی به این ابزار، مسیر زیر را طی نمایید:

Start -> Programs -> Accessories -> System Tools -> Disk Cleanup

۲-۳-٤- نتیجه‌گیری

در حالی که کاربران غیرحرفه‌ای در مواجهه با کاهش کارایی سیستم کامپیوتری خود، بلافاصله به فکر نصب مجدد سیستم عامل می‌افتند، کاربران حرفه‌ای این گزینه را در آخرین اولویت خود قرار می‌دهند. بدین ترتیب درحالی که ممکن است یک کاربر غیرحرفه‌ای در طول سال چندین بار سیستم‌عامل خود را نصب مجدد نماید، کاربر حرفه‌ای معمولاً این امر را هر چندسال یک‌بار انجام می‌دهد.

تنظیمات متعددی برای افزایش کارایی سیستم‌عامل وجود دارند، اما مهترین آن‌ها به ترتیب اهمیت، در این درس بیان گردید. برخی تنظیمات ممکن است ملموس به نظر نرسند و کاربر نتواند در رابطه با افزایش سرعت سیستم اظهار نظر دقیق بنماید، اما در بارکاری زیاد، این تفاوت متمایزتر خواهد شد. به عنوان مثال

اثر حذف عناصر غیرضروری در افزایش کارایی، از دید کاربر نهایی ملموس است، اما تنظیم حافظه‌ی مجازی، جز در بارکاری زیاد برای سیستم‌عامل، ملموس نیست.
برای اعمال شدن برخی تغییرات ممکن است نیاز به شروع مجدد[1] کار سیستم باشد، یا اینکه دست کم یک خروج[2] از کاربر کنونی انجام پذیرد. علت آن است که باید سیستم‌عامل، تخصیص منابع مانند حافظه‌ی اصلی را از نو انجام دهد.

۲-٤- تکلیف

۱. دو نمونه سرفصل‌های درس آزمایشگاه سیستم‌عامل ارائه شده در سایر دانشگاه‌های جهان را بیابید و ارائه دهید.
۲. مفهوم حافظه‌ی مجازی را به صورت علمی شرح دهید، کوچک بودن اندازه‌ی حافظه‌ی مجازی چه تأثیری بر عملکرد سیستم‌عامل دارد؟ اگر حافظه‌ی مجازی بسیار بزرگ باشد چه ایرادی دارد؟ توجه داشته باشید که تعریف ارائه شده در این آزمایش، تعریف علمی نیست و شما در پاسخ به این تکلیف نباید از مطالب آزمایش جاری استفاده کنید. بایستی اسم مرجع مورد استفاده در تعریف را نیز ذکر کنید.

۲-۵- مراجع

سلیمانی ، صادق. *افزایش کارایی کامپیوترهای شخصی*. تهران. انتشارات ناقوس. چاپ اول. پاییز ۸۳.

[1] Restart
[2] Logoff

۲-٦- دستورکار

توجه:
با توجه به اینکه اکثر تنظیمات ذکر شده در آزمایش‌ها، حساس و سیستمی هستند، مانند تغییرات در حافظه مجازی یا Registry، لذا *حتماً در حیطه آزمایش‌ها فعالیت انجام دهید*. هر گونه تغییر نابجا ممکن است موجب خرابی یا از دست رفتن ویندوز شود.

۱. به مسیر زیر بروید:

Start → run

با وارد کردن دستور msconfig و سپس زدن کلید Enter، از پنجره‌ایی که ظاهر می‌شود، سربرگ[1] Startup را انتخاب نمایید.

الف) نام دو مورد از برنامه‌هایی را که می‌شناسید، بنویسید.

ب) علامت وارسی[2] برنامه‌های غیرضروری را حذف نمایید.

ج) علامت وارسی چه برنامه‌هایی بهتر است حذف نشوند؟

د) از ستون Location، مکان عناصر را در رجیستری بیابید و یادداشت کنید سپس مسیر زیر را طی کنید:

Start → run → Regedit

پس از زدن کلید Enter در محیط باز شده و از طریق پنجره‌ی سمت چپ، آدرس ذکر شده در ستون Location را بیابید، اما از طریق رجیستری نسبت به حذف عناصر اقدام نکنید.

۲. ابتدا در Start menu بر Setting و سپس بر Control Panel کلیک کنید، بر شمایل[3] System کلیک دوگانه کنید و سربرگ General را انتخاب نمایید.

أ. کامپیوتر شما چقدر حافظه‌ی اصلی (RAM) دارد؟ میزان حافظه مجازی پیشنهادی چیست؟

ب. برای دستیابی به محل تنظیم حافظه مجازی، مسیر زیر را طی کنید:

در Start menu بر Setting و سپس بر Control Panel کلیک کنید، بر شمایل[4] System کلیک دوگانه کنید آنگاه سربرگ Advance را انتخاب

[1] Tab
[2] Check
[3] Icon
[4] Icon

نمایید و بر کلید ...Performance Option کلیک کنید. درایوی(یا درایوهایی) را که مایلید حافظه مجازی در آن قرار گیرد، مشخص کنید(اطمینان حاصل کنید که کلید Set را پس از تعیین مقدار برای هر درایو زده باشید).

چه گزینه‌هایی برای تنظیم حافظه مجازی وجود دارد؟ مقدار کمینه و بیشینه پیشنهادی ویندوز چقدر است؟

ج. به درایوی که حافظه‌ی مجازی اکنون در آن تنظیم شده است بروید و فایل متناظر با آن را که جزو فایل‌های سیستمی است و اکنون پنهان[1] است بیابید و اسم، پسوند و اندازه‌ی آن را بنویسید.

د. حافظه‌ی مجازی را طبق توصیه‌های ذکر شده در پیش‌آگاهی تنظیم نمایید و عملکرد را در گزارش‌کار یادداشت نمایید و به مربی نیز نشان دهید.

۳. بر درایو C از کامپیوتر خود کلیک راست نمایید و از Properties، سربرگ Tools را انتخاب نمایید و سپس از Error-Checking کلید ...Check Now را بزنید.

أ. گزینه‌های Check Disk را هرکدام در یک سطر شرح دهید.

ب. پس از انتخاب هر دو گزینه، Start را بزنید. آیا عملیات شروع می‌شود؟ در صورتیکه شروع نمی‌شود، پیام خطا چیست؟ علت را شرح دهید.

۴. پس از انتخاب Disk Cleanup برای یکی از درایوها، گزینه‌های موجود در آن برای حذف را فقط بنویسید (لازم به تشریح نیست).

۵. ترتیب انجام نرم‌افزارهای سودمند ویندوز (Scan Disk، Disk Defragment و Disk Cleanup) برای حصول بیشترین کارایی باید چگونه باشد. چرا؟

۶. به درایوی بروید که سیستم عامل در آنجا نصب شده است، این درایو معمولا درایو C است. از Tools به ...Folder Options بروید از گزینه‌ی View، امکان رویت فایل‌های نامریی را فعال سازید و گزینه‌ی Hide protected operating system files (Recommended) را از حالت انتخاب بردارید تا بتوانید فایل‌های سیستمی ویندوز را مشاهده نمایید. فایل سیستمی boot.ini را با Notepad باز کنید و سطر مشابه به زیر را بیابید و یک کپی مشابه به آن در زیر همین سطر ایجاد کنید اما پارامترهای پررنگ شده را در سطر کپی شده تغییر دهید و نتیجه را ذخیره و سیستم را Restart کنید.

[1] Hide

multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional" /fastdetect

سطر کپی شده

multi(0)disk(0)rdisk(0)partition(**2**)\WINDOWS="Microsoft Windows XP Professional**2**" /fastdetect

براساس خروجی ظاهر شده، در شروع به کار مجدد سیستم، عملکرد فایل boot.ini را بنویسید.

راهنمایی: با توجه اینکه در اکثر سایت‌های کامپیوتری از نرم‌افزارهای Freez بر کامپیوترها استفاده می‌شود، در زمان Restart نمودن سیستم، تغییرات به حالت اول برخواهند گشت. در این حالت توصیه می‌شود از سیستم‌عامل ویندوز تحت Virtual PC استفاده کنید.

آزمایش دوم

رفع‌عیب (Troubleshooting)

۳-۱- مقدمه

در این بخش به قابلیت‌هایی که سیستم‌عامل در جهت رسیدگی و رفع مشکلات متعدد روی داده از جانب کاربر، برنامه کاربردی یا خود سیستم عامل، فراهم می‌کند، پرداخته خواهد شد. شاید اغراق به نظر برسد اگر بیان شود که در این فصل آموخته خواهد شد که به ازای هر خطای[1] روی داده، اول علت چیست و دوم اینکه راه حل چیست! نکات مطرح شده، تجربیات حرفه‌ای محیط کار است که در عین حال، از بسیاری مفاهیم آموخته شده در درس سیستم‌عامل نیز پرده برخواهد داشت.

۳-۲- هدف

شناسایی رویدادها، هشدارها و خطاهای گزارش شده در سیستم‌عامل ویندوز و نحوه‌ی رسیدگی و رفع آن‌ها

[1] Error

۳-۳- پیش‌آگاهی

در سیستم‌عامل ویندوز برنامه‌های سودمند جانبی متعددی وجود دارد که ممکن است یک کاربر هرگز از آن‌ها استفاده نکند، اما برای کارشناسانی که از جنبه‌های مختلف نیاز به بررسی و ارزیابی دارند، مجموعه‌ای از ابزارها فراهم آمده است. یکی از این ابزارها که نقش بسیار مهمی در کشف و رفع خطاهای متعدد در سیستم‌عامل دارد، **Event Viewer** است که در Administrative Tools در Control Panel جای گرفته است.

با استفاده از Event Viewer می‌توان رویدادهای مربوط به سخت‌افزار، نرم‌افزار، سیستمی و امنیتی را جمع‌آوری نمود. این امکان در همه‌ی ویرایش‌های سیستم‌عامل ویندوز وجود دارد و در ویندوزهای کارگزار دارای گزینه‌های بیشتری است. علاوه بر این، نوع دیگری از رویداد این است که به دلایلی سیستم عامل شروع به فعالیت نکند و در همان آغاز شروع به کار[1] سیستم دچار مشکلاتی بشود. برای این مورد نیز راهکارهایی ارائه خواهد شد تا پوشش نسبتاً مناسبی از امکانات سیستم‌عامل برای رسیدگی و رفع خطاها داده شود.

۳-۳-۱- انواع رویدادها

پیش از پرداختن به دسته‌بندی‌های مختلف رویدادها در Event Viewer، به انواع آن اشاره می‌شود:

۱. خطا (Error): نشان از یک مشکل مهم مانند از دست رفتن داده یا از دست رفتن یک قابلیت دارد. به عنوان مثال اگر یک سرویس[2] در خلال شروع به کار سیستم، دچار مشکل گردد و اجرا نشود، یک رویداد خطا برای آن ثبت خواهد شد.

۲. هشدار(Warning): رویدادی که لزوما مهم به شمار نمی‌رود، اما می‌تواند علامت یک مشکل آتی باشد. برای مثال زمانی که فضای دیسک کم است (Low Disk Space)، یک رویداد از نوع هشدار ثبت می‌شود.

[1] Start up

[2] Service: سرویس یا خدمت، یک برنامه‌کاربردی است که در پیش‌زمینه (Background) ویندوز در حال اجرا است و رابط کاربری مستقیم ندارد و وظایفی از قبیل پایگاه داده، خدمات وب، تنظیمات خودکار شبکه و ... را برعهده دارد. برای انجام برخی تنظیمات بر سرویس‌ها و شناخت آن‌ها به آدرس Control Panel->Administrative Tools->Services مراجعه شود.

۳.  اطلاع(Information): رویدادی که اجرای موفق یک برنامه، گرداننده[1] یا سرویس را بیان می‌کند. برای مثال زمانی که گرداننده (درایور) کارت شبکه با موفقیت بارگذاری می‌شود و امکان استفاده از خدمات شبکه فراهم می‌شود، یک رویداد از نوع اطلاع تولید و ثبت می‌شود.

دو نوع رویداد دیگر نیز وجود دارند که به صورت پیش‌فرض فعال نیستند و در صورت انجام تنظیمات ویژه‌ایی (در Group Policy که در آزمایش‌های آتی به آن پرداخته خواهد شد) فعال می‌شوند، که عبارتند از:

۴.  رسیدگی موفقیت‌آمیز[2]: زمانی که یک فرایند امنیتی با موفقیت انجام می‌شود. مانند تلاش موفق یک کاربر برای ورود به سیستم، که به عنوان یک رسیدگی موفقیت‌آمیز ثبت خواهد شد.

۵.  رسیدگی ناموفق[3]: زمانی که یک فرایند امنیتی با موفقیت انجام نمی‌شود. برای مثال، وقتی که کاربری برای دسترسی به یک درایو در شبکه تلاش ناموفقی را انجام می‌دهد، یک رویداد از نوع رسیدگی ناموفق ثبت می‌شود.

۳-۳-۲- دسته‌بندی رویدادها

یک کامپیوتر با سیستم‌عامل ویندوز XP به‌صورت پیش‌فرض سه رده‌ی اصلی از رویدادها را ثبت[4] می‌کند:

۱. ثبت رویدادهای برنامه‌های کاربردی (Application Log)

مشتمل بر رویدادهای ثبت شده از برنامه‌های کاربردی یا سایر برنامه‌هاست. برای مثال یک برنامه‌ی پایگاه داده‌ها ممکن است در این دسته، رویدادی مربوط به خطای نوشتن در یک فایل را بنویسد.

۲. ثبت رویدادهای امنیتی (Security Log)

رویدادهایی از قبیل معتبر بودن یا نامعتبر بودن دسترسی یا استفاده از برخی منابع، در این رده ثبت می‌شوند.به عنوان مثال رویدادهای مربوط به ایجاد، حذف، بازکردن یا ... یک فایل در این

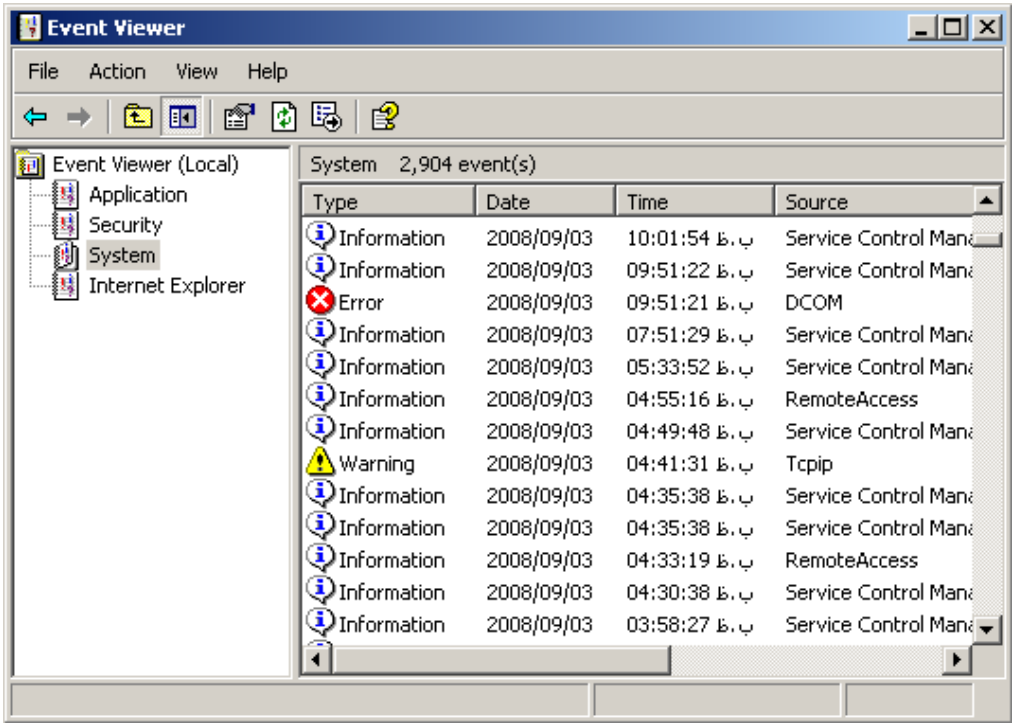
[1] Driver
[2] Successful Audit
[3] Failure Audit
[4] Log

بخش بایگانی می‌شوند. راهبر[1] می‌تواند تعیین کند که چه رویدادهایی در این بخش ثبت شوند. برای مثال می‌توان تنظیم کرد که رویدادهای ناموفق مربوط به تلاش برای ورود به سیستم همراه با نام کاربر و جزییات زمانی و ... به صورت دقیق ثبت شود.

۳. ثبت رویدادهای سیستمی (System Log)

رویدادهای مربوط به اجزای مختلف سیستم عامل در این بخش ثبت می‌شود. به عنوان مثال عدم بارگذاری یک گرداننده (درایور) یا یکی از اجزای سیستمی در خلال شروع به کار سیستم، در رویدادهای سیستمی ثبت می‌شود. این امر یا توسط خود سیستم عامل ثبت می‌شود یا توسط جزء مرتبط.

Event Viewer خود یک سرویس است که در زمان شروع به‌کار ویندوز، به صورت خودکار شروع می‌شود. همه‌ی کاربران امکان دسترسی به رویدادهای سیستمی و برنامه‌های کاربردی در Event Viewer دارند، اما رویدادهای امنیتی تنها برای کاربران با سطح دسترسی بالا (Administrators) قابل دسترسی است. در زیر نمایی از Event Viewer به همراه دسته‌بندی و برخی انواع رویدادها در ویندوز XP آمده است.



شکل ۳-۱ نمایی از Event Viewer

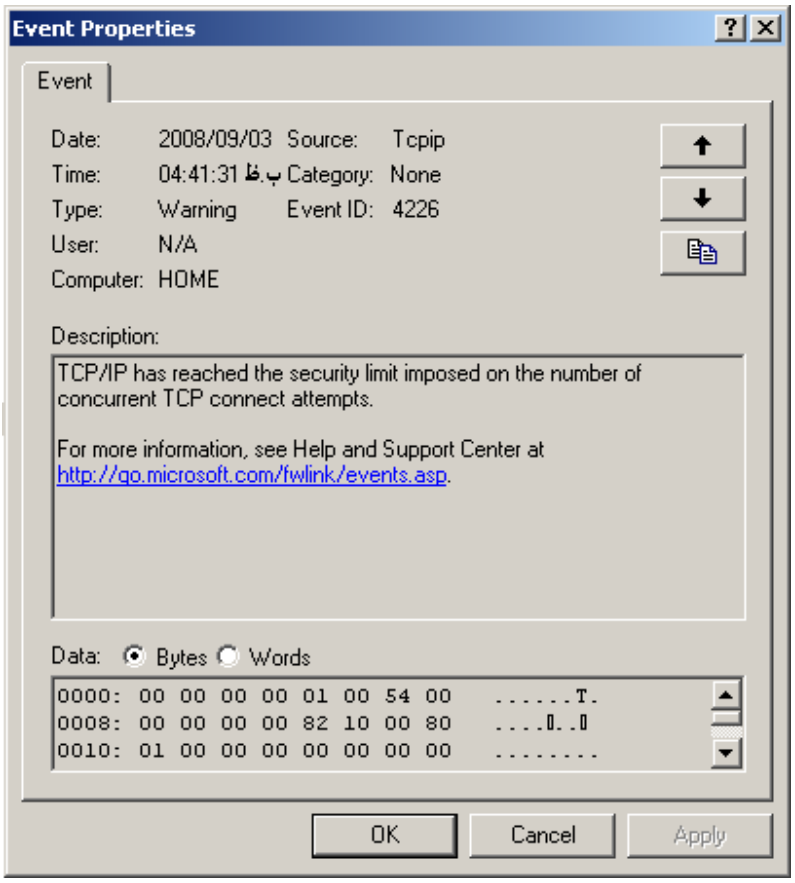
[1] Administrator: کاربری که بالاترین سطح دسترسی را در سیستم‌عامل داراست و تحت نام **راهبر** ترجمه شده است. اما در برخی کتاب‌ها، ترجمه‌ی **مدیر** را برای آن انتخاب کرده‌اند که در واقع این لفظ، ترجمه‌ی کلمه‌ی Manager است.

اما در سیستم‌عامل‌های کارگزار مانند Windows 2003 Server برخی رده‌های دیگر ثبت رویداد وجود دارد که در زیرآمده‌اند و که به جهت اختصار، در اینجا به آن‌ها پرداخته نمی‌شود.

- Directory service log
- File Replication service log
- DNS server log
- ...

۳-۳-۳- مشخصه‌های یک رویداد

رویدادهای موجود در سیستم‌عامل بسیار زیاد و متنوع هستند و درباره‌ی هر یک، مجموعه اطلاعاتی نگهداری می‌شود که می‌تواند آن رویداد را به صورت یکتا در بین هزاران رویداد دیگر متمایز کند و به اهل فن امکان تبادل و رسیدگی به آن اطلاعات را بدهد. پس از کلیک دوگانه[1] بر یک رویداد، پنجره‌ی زیر ظاهر می‌شود، که در ادامه به شرح تک تک اجزای آن پرداخته خواهد شد.



شکل ۳-۲ مشخصه‌های یک رویداد

[1] Double Click

جدول ۳-۱ تشریح مشخصه‌های رویداد

فیلد	معنا
Date	تاریخ اتفاق افتادن رویداد
Time	ساعت محلی سیستمی که رویداد در آن زمان در آن اتفاق افتاده است
User	نام کاربری که رویداد از جانب وی واقع شده است. در صورتی که رویداد از طرف فرایندی در کامپیوتر کارگزار اتفاق افتاده باشد یا به نوعی وابسته به شخص نباشد به جای نام کاربر Client ID یا ID اصلی فرایند مسبب رویداد ثبت خواهد شد. این امر بیشتر زمانی اتفاق می‌افتد که ویندوز به یک فرایند اجازه‌ی دستکاری امنیتی در فرایندی دیگر را بدهد.
Computer	نام کامپیوتری که رویداد در آن واقع شده است. معمولاً اسم کامپیوتر جاری است، مگر اینکه در حال بررسی Event Viewer کامپیوتر دیگری از راه دور باشیم.
Event ID	شماره‌ی متمایز کننده‌ی رویداد. سطر اول Description معمولا با نام رویداد همراه است. به عنوان مثال شماره‌ی ۶۰۰۵، مربوط به رویداد شروع به کار Event Viewer است. که سطر اول Description متناظر با آن ".The Event log service was started" خواهد بود. Event ID به همراه Source، جهت مشخص کردن یکتای رویداد در مواردی مانند پشتیبانی، مورد استفاده قرار می‌گیرد.
Source	نرم‌افزاری که رویداد را ثبت می‌کند، که می‌تواند نام یک برنامه مانند "SQL Server" باشد یا جزیی از سیستم یا نام یک گرداننده. برای مثال "Elnkii" بیانگر گرداننده‌ی کارت شبکه یا EtherLink II driver است.
Type	رده‌بندی امنیتی رویداد را مشخص می‌کند: خطا، هشدار یا اطلاع در رویدادهای ثبت شده‌ی برنامه‌های کاربردی یا سیستمی و رسیدگی موفقیت‌آمیز یا ناموفق در رویدادهای امنیتی.
Category	دسته‌بندی رویداد بر اساس منبع ایجاد کننده‌ی آن. اولین استفاده‌ی این اطلاعات در داده‌های ثبت شده‌ی امنیتی است.
Description	شرح مختصر رویداد از دیدگاه شرکت مایکروسافت به‌علاوه آدرس اینترنتی احتمالی برای کسب اطلاعات بیشتر. اغلب اوقات، درک چنین شرحی نیاز به دانش تخصصی در زمینه‌ی رویداد دارد.
Data	بیانگر وضعیت بخشی از حافظه‌ی اصلی، که خطا در زمان اتفاق افتادن رویداد در آن بخش گزارش شده است. چنین داده‌ایی در دوقالب Byte و Word نمایش داده می‌شود و به‌خصوص برای بررسی‌های سیستمی دقیق‌تر قابل استفاده است.

۳-۳-٤- مشکلات مربوط به پیش از شروع به کار سیستم‌عامل

گاهی ممکن است بر اثر تغییرات ایجاد شده در سیستم، نصب یک برنامه‌ی جدید یا حذف برخی برنامه‌ها یا فایل‌های سیستمی، دیگر امکان شروع به کار مجدد سیستم‌عامل وجود نداشته باشد. به عنوان مثال، ممکن است بر اثر نصب یک سخت‌افزار جدید و درایور آن، پس از شروع به کار مجدد سیستم و پیش از بالا آمدن ویندوز، خطایی مبنی بر عدم امکان ادامه‌ی صحیح فعالیت سیستم‌عامل بروز کند. در این صورت دیگر امکان دسترسی به Event Viewer برای بررسی دلایل و چاره‌ی خطا وجود نخواهد داشت. در این حالت می‌توان از وضعیت اضطراری برای اجرای ویندوز که به حالت امن یا Safe Mode موسوم است، بهره برد.

Safe Mode -۱-٤-۳-۳

Safe Mode یا حالت امن، حالتی از اجرای ویندوز است که با حداقل فایل‌های ضروری صورت می‌گیرد. بدین ترتیب دو مزیت عمده ایجاد خواهد شد، اول اینکه اگر برنامه‌ایی به تازگی نصب شده و سبب ایجاد ناسازگاری در سیستم شده است، در این حالت به حافظه‌ی اصلی بارگذاری نمی‌شود و اجرا نمی‌گردد و موجب جلوگیری از عملکرد سیستم‌عامل ویندوز نمی‌شود و دوم اینکه، حتی اگر اجزایی از خود سیستم‌عامل نیز آسیب دیده باشند، در صورتی‌که آن اجزا خیلی مهم و حساس نباشند، باز هم امکان ادامه‌ی کار با سیستم‌عامل ویندوز و جبران خرابی وجود دارد. لازم به ذکر است که Safe Mode تنها مختص به سیستم‌عامل ویندوز نیست و سایر سیستم‌عامل‌ها از جمله لینوکس نیز مکانیزم‌های مشابهی برای کار با سیستم در مواقع بحرانی دارند.

وارد شدن به حالت Safe Mode برای بیشتر سخت‌افزارها با فشردن کلید F8 در زمان روشن شدن سیستم و ظاهر شدن متن‌های مربوط به شروع به کار سیستم و BIOS است. اغلب در این حین صدای بوق POST یا تست اولیه و راه‌اندازی سیستم نیز شنیده می‌شود. همچنین ممکن است در این زمان از کاربر خواسته شود تا دستگاهی را که مایل است شروع به کار سیستم از آن صورت گیرد، انتخاب کند، به عنوان مثال: Floppy Drive، Hard Disk یا CD ROM که شما باید Hard Disk را انتخاب نمایید.

لازم به توجه است که چون استفاده از Safe Mode توسط افراد حرفه‌ایی صورت می‌گیرد، بسیاری از اموری که در حالت عادی اجرای ویندوز به عنوان عملیات حفاظت شده تلقی می‌شدند و امکان پذیر نبودند، در Safe Mode امکان‌پذیر می‌شود، مثلا حذف یا جایگزینی فایل‌های سیستمی. بنابراین باید عملیات اینچنینی با آگاهی و دقت صورت پذیرد.

برخی گزینه‌های Safe Mode در زیر آمده است که ممکن است بر حسب نسخه سیستم‌عامل یا نوع سخت‌افزار، بعضی از آن‌ها در کامپیوتر شما ظاهر نشود:

1. Safe Mode
2. Safe Mode with Networking
3. Safe Mode with Command Prompt
4. Enable Boot Logging
5. Enable VGA Mode
6. Last Known Good Configuration
7. Directory Services Restore Mode
8. Debugging Mode
9. Disable Automatic Restart on System Failure
10. Start Windows Normally
11. Reboot
12. Return to OS Choice Menu

بعضی از موارد فوق در اینجا تشریح خواهد شود و برخی دیگر نیز به عنوان تکلیف خواهد آمد، به عنوان مثال Safe Mode with Networking زمانی مورد استفاده قرار می‌گیرد که لازم باشد در Safe Mode از امکانات شبکه نیز استفاده برد یا Last Known Good Configuration سبب خواهد شد که سیستم به آخرین وضعیت پایدار پیشین برگردد. استفاده از این گزینه حتی ممکن است سبب شود که برخی درایورهای نصب شده نیز از حالت نصب خارج شوند، زیرا به صورت کامل، با سیستم عامل سازگار نبوده‌اند.

BIOS -۲-٤-۳-۳

BIOS که بررسی آن خود نیاز به یک مجال مفصل دارد، یک برنامه‌ی کوچک است که بر حافظه‌ای فقط خواندنی[1] قرار گرفته و دو وظیفه‌ی اصلی تست و مقداردهی اولیه‌ی سخت‌افزار و بارگذاری هسته‌ی سیستم‌عامل در حافظه‌ی اصلی را بر عهده دارد. در تست اولیه، سخت‌افزارهای عمده‌ی سیستم مانند CPU، RAM، کارت گرافیکی و ... مورد بررسی قرار می‌گیرند و در صورت استقرار صحیح و بدون خطا، مقداردهی اولیه‌ی آن‌ها صورت می‌پذیرد و سپس آدرس محل ذخیره‌ی سیستم عامل از قطاع شماره‌ی صفر حافظه‌ی جانبی (دیسک سخت) خوانده شده و از آن آدرس، بارگذاری جزء اصلی سیستم‌عامل یعنی هسته‌ی سیستم‌عامل، به حافظه‌ی اصلی صورت پذیرفته و کنترل اجرای دستورات به هسته‌ی سیستم‌عامل که اکنون در RAM یا حافظه‌ی اصلی مستقر است، منتقل خواهد شد.

BIOS همچنین داده‌های حساسی مانند ساعت و تاریخ سیستم را در زمان خاموشی کامپیوتر نگهداری می‌کند و برای این کار از باتری قرارگرفته بر Motherboard کمک می‌گیرد.

[1] ROM

نقش BIOS در عیب‌یابی، به خصوص در زمانی است که خطا مربوط به پیش از بارگذاری سیستم‌عامل باشد. علاوه بر اینکه تنظیمات سخت‌افزارهای مختلف در BIOS ثبت می‌شود، در صورت ایراد سخت‌افزاری عمده، BIOS شروع به ایجاد بوق‌های متنوع بسته به نوع خطا خواهد کرد. به عنوان مثال، اگر در هنگام روشن شدن سیستم، بوق ممتد شنیده شود، بیانگر ایراد سخت‌افزاری یا درست جانیفتادن RAM در شکاف مربوطه‌اش است.

۳-۳-۵- تفسیر رویداد و رفع عیب آن

اولین گام در رفع‌عیب، آگاهی از منبع و علل بروز خطاست. بدین ترتیب باید همانند یک پزشک که پیش از تجویز نسخه، باید اطلاعات کاملی از بیمار و علایم بیماری کسب نماید، برای یک مهندس کامپیوتر نیز پیش از پرداختن به راه‌حل، باید اطلاعات مناسبی در اختیار قرار داده شود که این مورد را می‌توان از Event Viewer کسب نمود.

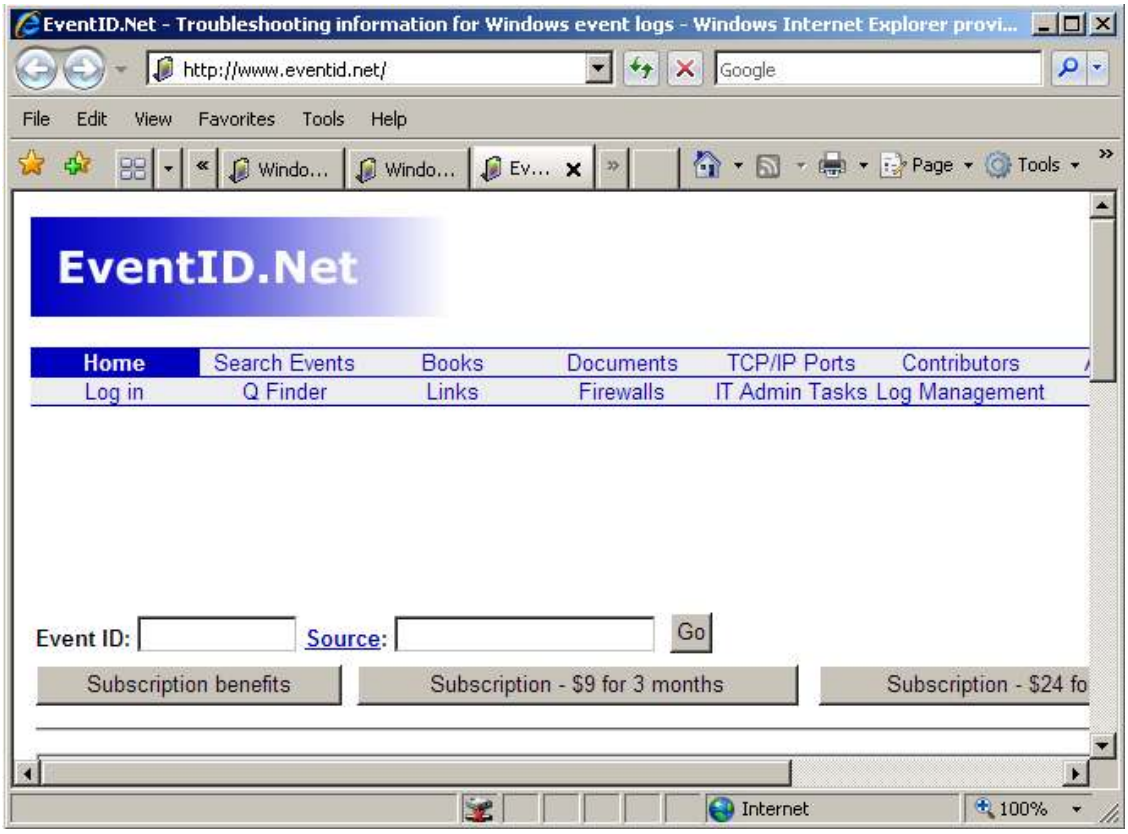
گرچه ویندوز در این زمینه اطلاعات مفیدی از طریق فیلد Description در مشخصه‌ی رویداد عرضه می‌کند، اما این داده‌ها معمولا برای کاربران غیرحرفه‌ای و تازه‌کار، ثقیل و پیچیده است. از طرفی مراجعه به پایگاه داده‌ی برخط مایکروسافت نیز که پیوند آن در بخش آخر فیلد Description تحت عنوان اطلاعات بیشتر می‌آید، اگر جهت‌دار و با آگاهی نباشد، بیشتر کاربر را سردرگم خواهد کرد.

در محیط کار، بهره بردن از تجربه‌ی افراد خبره در زمینه‌ی رسیدگی به خطاهای متعدد سیستم‌عامل، امری بسیار سودمند و سریع برای دستیابی به راه‌حل درست است. البته این مورد نیز به سادگی در دسترس نیست، اما خوشبختانه وب‌سایتی به آدرس **www.EventID.net** که مجموعه نظرات کاربران را به صورت دسته‌بندی شده در اختیار قرار می‌دهد، راهگشای مناسبی در این زمینه به شمار می‌رود. برای یافتن تجارب دیگر افراد خبره درباره‌ی رویداد مورد نظر، کافی است پس از رفتن به آدرس مورد نظر، Event ID و Source رویداد مربوطه را استخراج کرده و در مکان مناسب نوشت (شکل ۳-۳). سپس در صفحه‌ی ظاهر شده بر پیوندی مشابه Comments and links for event id … from source … کلیک شود.

لازم به ذکر است که برای یک خطا ممکن است دلایل مختلفی وجود داشته باشد و در نهایت، این کاربر است که تصمیم می‌گیرد، کدام راهکار پیشنهاد شده را به کار ببرد و در نهایت نیز وی مسئول عواقب بعدی خواهد بود.

البته راهکار استفاده از وب‌سایت www.EventID.net همیشه راهگشا نیست و برای یافتن اطلاعات برخی خطاهای حساس، به ثبت‌نام و پرداخت هزینه‌ایی در این زمینه، به وب‌سایت مذکور نیاز است، که

بنابرمحدودیت‌های کاربران ایرانی در زمینه‌ی پرداخت الکترونیکی، راهکارهای دیگری برای این امر پیشنهاد می‌شود. Google Groups از گروههای اینترنتی معتبر و بزرگ است که بسیاری از مسایل تخصصی و فنی و علوم رایانه را دربر می‌گیرد و بحث‌های مفیدی در آن یافت می‌شود.



۳-۳ صفحه اول وب‌سایت www.EventID.net

کافی است Event ID، Source و چند کلمه‌ی کلیدی از Description رویداد را در آن نوشت و کلید جستجو را زد و بعد با دقت در بین سوال و جواب‌های یافت شده، مورد مناسب را پیدا کرد و راهکارهای پیشنهادی را پس از تأمل، به کار گرفت.

۳-۳-۵-۱- جستجو در اینترنت

برکسی پوشیده نیست که یافتن سریع مستندات مورد نیاز در بین میلیون‌ها صفحه‌ی وب، به مهارت در جستجو نیاز دارد. به همین خاطر افراد خبره، مطلبی را که ممکن است یک تازه کار پس از ساعت‌ها جستجو در اینترنت به سختی بیابد، با یک جستجوی سریع می‌یابند. جهت پیشرفت سریع در جستجو برای

یافتن علت یک رویداد در سیستم، بهترین پیشنهاد این است که پرسش خود را مانند یک انگلیسی زبان مطرح کنید و ترتیب کلمات را به نحوی که در زبان انگلیسی متعارف‌تر است وارد کنید. بدین ترتیب بدیهی است که تسلط بر انگلیسی به یافتن سریعتر جواب کمک خواهد کرد.

لازم به توجه است، حتی زمانی که هنوز سیستم‌عامل بارگذاری نشده است و پیام خطایی بر زمینه‌ی سیاه صفحه نمایش ظاهر می‌شود نیز، می‌توان عبارات آن پیام را در کامپیوتر دیگری که به اینترنت متصل است جستجو کرد و راهکار مناسب را یافت.

۳-۳-۵-۲- استفاده از کتاب‌های مرجع

کتاب‌های متعددی در زمینه‌ی رفع عیب در سیستم‌عامل تألیف شده‌اند، که برخی از آن‌ها جامع هستند و حالت مرجع دارند و معمولاً در کتابخانه‌های شرکت‌های مرتبط با فناوری اطلاعات شناخته شده و مورد استفاده هستند. یک مورد از این کتاب‌ها، مرجعی است که در بخش مراجع این آزمایش به آن اشاره شده است.

۳-۳-۵-۳- بهره بردن از تجربیات افراد خبره

تجربه امری است که در فرایند عیب‌یابی و رفع آن نقش کلیدی بازی می‌کند. افراد خبره همانند پزشکان ماهر که با چند سوال کلیدی، بسیاری از حدسیات را کنار می‌زنند عمل می‌کنند و از هدر رفتن زمان و تلاش جلوگیری می‌کنند.

بی‌شک مراجعه به یک فرد خبره یا در اختیار داشتن یک فرد با تجربه، روند رسیدگی به بسیاری از مشکلات سیستم‌عامل را تسهیل و تسریع خواهد کرد، اما با توجه به اینکه این امکان معمولاً فراهم نیست، می‌توان از نظرات مستند شده‌ی آن‌ها که در اینترنت گذارده شده است، استفاده کرد.

۳-٤- نتیجه‌گیری

بررسی متناوب خطا در بسیاری سازمان‌ها جزو وظایف کارشناسان شبکه به حساب می‌آید. یافتن علت خطا در زمان مناسب ممکن است از بسیاری زیان‌های محتمل جلوگیری کند، به عنوان مثال خطای دیسک یعنی اینکه به زودی دیسک درایو مستهلک شده، از کار خواهد افتاد. درصورتی که کارشناس، هر روز Event Viewer کامپیوترهای کارگزار حساس را بررسی کند، پیش‌گیری از بسیاری رویدادهای خطرناک میسر خواهد شد.

یک کارشناس کامپیوتر نیز با داشتن درک کافی از سیستم‌عامل و امکانات آن برای رفع عیب، چه در استفاده‌های شخصی و چه در محیط کار، فعالیت‌های پرثمرتری خواهد داشت.

۳-۵- تکلیف جلسه بعد

۱. کاربرد موارد زیر از گزینه‌های Safe Mode را به صورت دقیق بیان کنید، یعنی اینکه بنویسید هرکدام در چه موقعیتی و برای چه منظوری مورد استفاده قرار می‌گیرد:

a) Safe Mode with Command Prompt
b) Enable Boot Logging
c) Enable VGA Mode
d) Directory Services Restore Mode
e) Debugging Mode
f) Disable Automatic Restart on System Failure

۲. خطای با Event ID: 22 و Source: Norton Antivirus را از طریق اینترنت بیابید و ضمن بیان نوع و عنوان رویداد، دو نظرارائه شده در رابطه با آن را (به فارسی) بنویسید.

۳. معنی بوق‌های بایوس برای خطاها و مشکلات مختلف سخت‌افزاری را بیابید.

۳-۶- مراجع

1. C. Wolf, **Troubleshooting Microsoft Technologies: The Ultimate Administrator's Repair Manual**, Addison-Wesley Pub., 2003.

۳-۷- دستور کار

توجه: از پشت برگه‌ی گزارش کار نیز می‌توانید برای نوشتن گزارش استفاده نمایید.

۱. ابتدا به Event Viewer که در Administrative Tools در Control Panel قرار دارد بروید و با کلیک راست بر System و انتخاب گزینه‌ی Clear All Events آن را خالی کنید. سپس به Services که در Administrative Tools در Control Panel قرار دارد بروید و سرویس DNS Client را یافته و با کلیک راست بر آن، گزینه‌ی Restart را انتخاب نمایید.
حال به System در Event Viewer برگردید. چند رویداد ثبت شده است؟ به ترتیب ذکر کنید که هرکدام چه اطلاعی می‌دهند.

۲. به Start سپس Run بروید و عبارت gpedit.msc را تایپ نمایید، سپس OK را کلیک کنید. در محیط باز شده، از Computer Configuration، Windows Settings و بعد Security Settings و سپس از Local Policies گزینه‌ی Audit Policy را انتخاب نمایید و در پنجره‌ی سمت راست گزینه‌ی Audit logon events را کلیک دوگانه کنید.
این گزینه برای ثبت تلاش‌های موفق یا ناموفق ورود به سیستم، کاربرد دارد.
حال از پنجره‌ی باز شده، هر دو مورد Success و Failure را علامت بزنید. سپس سیستم را Logoff کنید و در ورود مجدد به سیستم، ابتدا یک نام کاربری و رمزعبور ناشناخته وارد کنید تا پیام خطا ظاهر شود و سپس نام کاربری و رمزعبور مجاز برای ورود به سیستم را وارد کنید.
الف) تصور می‌کنید اکنون باید در کدام بخش از Event Viewer به دنبال رویداد ثبت‌شده‌ی ورود موفق و ناموفق بگردید؟
ب) رویداد تلاش ناموفق برای ورود را بیابید و شرح آن را در برگه‌ی گزارش کار بنویسید.
توجه:
چنانچه به دلیل نرم‌افزار خاص نصب شده در سایت، پس از Login مجدد، تمام تنظیمات به حالت اول برگشته بود، به جای انجام Logoff از Switch User استفاده نمایید.

۳. در Event Viewer به منوی View بروید و سپس گزینه‌ی ...Find را انتخاب نمایید. از لیست پایین کشیدنی Event Source گزینه‌ی Application را انتخاب کنید و سپس تمام موارد موجود

در لیست پایین کشیدنی Category را بنویسید. وجود همه‌ی این موارد در لیست Category به چه معنی است؟

۴. دو رویداد خطای دلخواه را در یکی از رده‌های Application، Security یا System بیابید و علاوه بر یادداشت Event ID و Source شرح (Description) آن‌ها را یادداشت کنید. تلاش کنید خطایی را انتخاب کنید که مضمون آن را می‌فهمید.

۵. اگر دیسک سخت به تازگی صدای زیادی ایجاد نماید و در Event Viewer با رویداد زیر مواجه شوید:

Source	Disk
Type	Error
Description	The device, <device>, has a bad block.

و در ارجاع به وب‌سایت www.EventID.net توصیه‌های زیر را ببینید:

Event ID: 7	
Source	Disk
Type	Error
Description	The device, <device>, has a bad block.
English please!	Request a translation of the event description in plain English! An example of "English please" is available here.
Comments	Mihai Andrei (Last update 2/28/2006): This event might be logged when you format a logical disk on an extended partition on an ATAPI hard disk drive on a Microsoft Windows NT 4.0-based computer. See M817755 for a hotfix applicable to Microsoft Windows NT. See MSW2KDB and "Lacie Support FAQ" for additional information about this event. Ionut Marin (Last update 10/6/2004): As per Microsoft: "The device has a bad block of memory, which Windows attempted to read. The data might be missing or corrupted". See the link to "EventID 7 from source Cdrom" for details on this event. Adrian Grigorof According to Microsoft: "From the command prompt, run the Chkdsk utility with the /r option on the named partition. If Chkdsk reports no errors but you continue to receive this message, run hardware diagnostics on the disk drive named in the message and on its controller. You might need to contact the vendor of the device for technical support, and you might have to replace the controller, the disk drive, or both."

	Fibtech According to Microsoft you need to translate the hard disk number to a physical drive on the system. You can determine the hard disk number by looking in the registry, but you need to know if you are using IDE drives, SCSI drives or a combination of the two. Disk Administrator will display the drives in the order they are enumerated on each controller and in the order that the controller device drivers are loaded. If you are using multiple controllers, the order in which they are identified is based on I/O port and controller BIOS address assignments. See M159865 for more details.
Links	M159865, M817755, EventID 7 from source Cdrom, Lacie Support FAQ, MSW2KDB
Search	Google Web - Microsoft Support - Microsoft Search - Google Microsoft - EventID.Net Queue - More links...
Various	Send comments - Notify me when updated

به ترتیب، چه راهکارهایی را برای رفع عیب پیشنهاد می‌کنید.

۶. سیستم خود را Restart کنید و بیان کنید،

الف) بایوس آن ساخت چه شرکتی است و با زدن چه کلیدی می‌توان به بایوس وارد شد؟

ب) همچنین گزینه‌های مختلف مرتبط به boot شدن را در بایوس بیابید و در برگه‌ی گزارش‌کار یادداشت کنید.

ج) آیا می‌توانید با تغییراتی در بایوس کار کنید که بتوان انتخاب کرد که هر زمان سیستم شروع به کار کرد، کلید Numlock روشن باشد یا خاموش؟ مکان تنظیم را در صورت وجود بنویسید.

آزمایش سوم

نظارت بر کارایی سیستم‌عامل

٤-١- مقدمه

پارامترهای متعددی در سیستم‌عامل وجود دارد که می‌توان هر یک را به صورت مجزا مورد نظارت و ارزیابی قرار داد. بدین ترتیب علاوه بر درک مفاهیم نظری آموخته شده در درس سیستم‌عامل، به صورت عملی نیز می‌توان هریک از موارد آموخته شده را به کار گرفت.

در این فصل، پس از تقسیم گلوگاه‌های اصلی عملکرد سیستم عامل در چهار رده‌ی اساسی، به مهمترین مفاهیم مرتبط با آن‌ها و نحوه‌ی اندازه‌گیری، مقایسه و نظارت بر عملکردشان پرداخته خواهد شد.

٤-٢- هدف

آشنایی با نحوه‌ی نظارت، اندازه‌گیری و ثبت آمارهای مربوط به اجزای مختلف سیستم‌عامل ویندوز

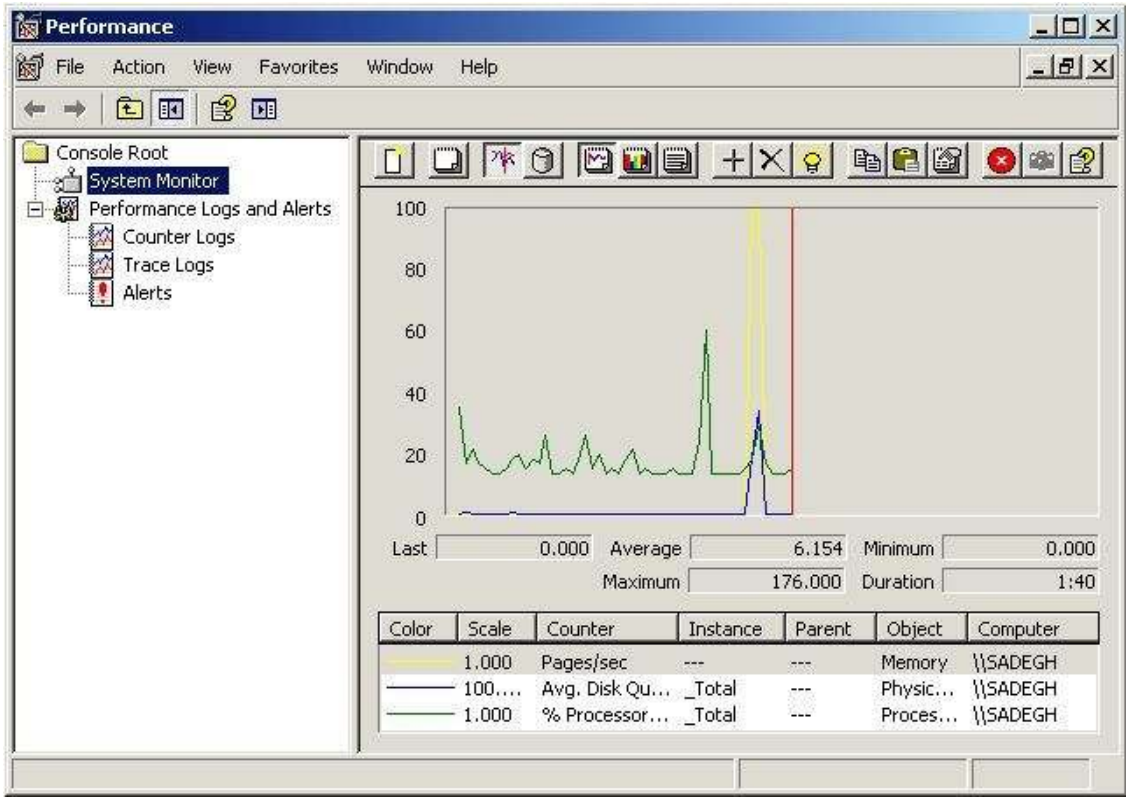
٤-۳- پیش‌آگاهی

جمله‌ی معروفی وجود دارد که اگر نتوانید چیزی را اندازه بگیرید، نمی‌توانید آن را تنظیم کنید. تمام جزئیات مربوط به سیستم عامل ویندوز و آنچه در مورد کنترل و نظارت منابع، مورد نیاز است در performance monitor موجود است. به عنوان مثال می‌توان دانست که تعداد Page Faultهای مربوط به مدیریت حافظه، در یک بازه ی زمانی چندبار بوده است.

٤-۳-۱- آشنایی با Performance Monitor

Performance Monitor برنامه‌ایی تعبیه شده در ویندوز است که می‌توان آمارهای مختلفی را در رابطه با اجزای مختلف سیستم‌عامل، به کمک آن کسب نمود. دو جزء اصلی Performance Monitor عبارتند از:

۱. **System Monitor**
۲. **Performance Log and Alert**



که در شکل فوق در بخش سمت چپ پنجره آمده‌اند. در زمان انتخاب هر یک، تنظیمات و محتویات آن در بخش سمت راست، ظاهر خواهد شد. گزینه‌ی اول مربوط به رویت آمارهای متنوع کنونی در رابطه با عملکرد پارامترهای مختلف سیستم عامل است و گزینه‌ی دوم، در رابطه با ثبت آمارهای پارامترهای مختلف سیستم‌عامل به شیوه‌های متنوع، در بازه‌های زمانی مشخص و انجام عملی در صورت بروز رویدادی در سیستم است. به عنوان مثال در زیر بخش Alerts از بخش Performance Log and Alerts، می‌توان تنظیماتی انجام داد که اگر بارکاری پردازنده[1]، از ۵۰ درصد بالاتر برود، پیام یا ایمیلی به کامپیوتری ارسال شود یا در جایی ثبت شود.

برای آشنایی دقیق و استفاده‌ی مناسب پارامترهای سیستم‌عامل، لازم است ابتدا به سه مفهوم رایج در Performance Monitor پرداخته شود:

۱. **شیء(Object) :**هر جزء سیستم که دارای برخی ویژگی‌های قابل اندازه‌گیری باشد. Object یا شیء می‌تواند یک جزء فیزیکی سیستم (مانند CPU یا حافظه)، یک جزء منطقی (مانند یک disk volume) یا یک عنصر نرم‌افزاری (مانند یک process یا thread) باشد.

۲. **نمونه (Instance):** نشان‌دهنده تعداد تکرارهای موجود یک شی در سیستم است. به عنوان مثال ممکن است در یک سیستم، دو کارت شبکه داشته باشیم، پس دو instance از Network interface خواهیم داشت.

۳. **شمارنده(Counter)**: یک ویژگی قابل اندازه‌گیری از شیء را نشان می‌دهد. مانند شیء Processor که دارای شمارنده‌های متعددی است؛ درصد زمان مشغول بودن آن، درصد زمانی که در modeهای Privilege و User بوده است، چند مورد از شمارنده‌های آن به حساب می‌آیند.

لازم به ذکر است که در نوشتن ابتدا نام شیء می‌آید و سپس نام شمارنده. به عنوان مثال: شمارنده‌ی مربوط به درصد زمان‌هایی را که برای نوشتن در دیسک مصرف شده، به فرم زیر می‌نویسند:

Physical Disk\Disk Write Time

[1] CPU Load

ویرایش‌های مختلف سیستم‌عامل‌های ویندوز دارای صدها شمارنده هستند. مانند تعداد Packetهای انتقالی در واحد ثانیه، درصد زمان CPU برای Modeهای مختلف و ...

٤-۳-۲- انواع خروجی‌های System Monitor

Performance Monitor جهت پوشش بررسی‌های مختلف در رابطه با سیستم عامل، خروجی‌های مختلفی را برای کاربر ارائه می‌دهد که در ادامه به صورت خلاصه به آن‌ها اشاره شده است:

۱. **Graph**: این خروجی که حالت پیش‌فرض نیز هست، نشان دهنده‌ی وضعیت کنونی در قالبی شبیه به نمودارهای گرافی متداول است و به خصوص برای رویت تغییرات نهایی در مقدار شمارنده‌ها مفید است.
۲. **Histogram**: این نمودار مقادیر کنونی شمارنده‌ها را به صورت میله‌ای نشان می‌دهد و برای مقایسه بین یک شمارنده در چند کامپیوتر مناسب است، مثلاً زمان مشغول بودن CPU در چندین Server را می‌توان از این طریق با هم مقایسه نمود.
۳. **Report**: مقادیر کنونی اشیاء و شمارنده‌ها را به صورت عددی اعشاری نمایش می‌دهد و برای به دست آوردن مقدار دقیق استفاده می‌شود. بدین ترتیب که در هر لحظه مقدار کنونی شمارنده را در دسترس قرار می‌دهد ولی می‌توان مقدار متوسط یک شمارنده را در بازه‌ایی از زمان نیز با آن مد نظر داشت.

برای مشاهده و استفاده از این نمودارها، کافی است در نوار ابزار بر شمایل‌های زیر کلیک نمایید:

٤-۳-۳- گلوگاه‌های کارایی در سیستم‌عامل

چهار منبع مهم گلوگاه کارایی که در بررسی عملکرد و وضعیت سیستم‌عامل نقش اساسی دارند عبارتند از:

۱. **Memory**
۲. **Disk Subsystem**
۳. **Network card and software**
۴. **Processor**

معمولاً در استفاده‌های عملی از Performance Monitor، این چهار گزینه به کرات مورد بررسی قرار می‌گیرند، که هرکدام دارای شمارنده‌های متعددی هستند. به عنوان مثال در رابطه با شیء Processor ، User Time به درصد زمانی از CPU اشاره دارد که برای کاربر اختصاص یافته است. یا اگر اغلب اوقات درصد زمان CPU(CPU Time) بیش از ۷۵٪ باشد، نشان دهنده‌ی آن است که پردازنده تقریباً به سختی کار می‌کند و بهتر است که رایانه‌ی شخصی یا Server از یک پردازنده سریع‌تر یا از یک پردازنده اضافی کمک بگیرد. همچنین درصد زمان وقفه (Interrupt Time)، به شما می‌گوید که چه زمانی از پردازنده برای سوییچ به وضعیتی که کار دیگری را (که از طرف سخت‌افزار مانند کارت شبکه، کارت گرافیکی، صفحه کلید و ... تقاضا شده) انجام دهد، صرف شده است.

٤-٣-٤- بایگانی داده‌های کارایی[1]

دانستن کارایی کنونی Server مفید است، اما دانستن کارایی پیشین آن (یک هفته یا یک ماه قبل) نیز اهمیت دارد، زیرا از روی آن می‌توان به دلایل تغییر کارایی کنونی پی برد. بخش دوم از Performance Monitor که دارای حساسیت زیادی است، Performance Log and Alert می‌باشد. در این بخش امکان ثبت داده‌های کارایی در فایل‌های بایگانی[2] وجود دارد. اطلاعات logمی‌توانند در یک فایل متنی با کاما از هم جدا شده باشند یا با کاراکتر Tab. که این امر قابلیت بازکردن آن‌ها در محیطی مانند Excel برای انجام تحلیل‌های آماری را خواهد داد، یا حتی می‌توانند در قالب فایل‌های Microsoft SQL نوشته شوند، تا بتوان برنامه‌های کاربردی برای تحلیل و پردازش آن‌ها ایجاد کرد.

Performance Log and Alert سه نوع Log را حمایت می‌کند:

۱. Counter Log
۲. Trace Log
۳. Alert Log

Counter Log:

داده‌های کامپیوتر کنونی[3] یا دوردست[4] درباره میزان استفاده از سخت‌افزار و فعالیت Serviceها را (نمونه برداری و) در بازه‌هایی اززمان که برایش مشخص شده، ثبت می‌کند.

Trace Log:

[1] Logging Performance Data
[2] Log File
[3] local
[4] remote

مبتنی بر رویداد[1] است. زیرا برخی داده‌ها مانند I/O دیسک و page Faultها ممکن است در یک بازه‌ی زمانی اصلاً اتفاق نیفتند. پس بهتر است به سیستم عامل گفته شود، هرگاه که اتفاق افتادند ثبت شود و سیستم عامل خود را محدود به یک بازه‌ی زمانی محدود ننماید.بنابراین مقادیر شمارنده‌های ذکر شده که مربوط به دیسک بودند، در هنگام رویداد، در فایل Log ذخیره می‌گردند.

Alert Log:

وظیفه‌ی اصلی این بخش نظارت بر برخی شمارنده‌های اشیاء و انتظار برای از حد گذشتن مبناهایی که شما تعیین کرده‌اید، است. یک Alert Log را برای اینکه کاری را به فرد یادآوری کند، تنظیم می‌کنیم، مانند اجرای یک برنامه یا ارسال یک پیام در زمان رویدادی همچون عبور میزان استفاده‌ی CPU از ۹۰٪.

٤-٤- مراجع

M. Minasi, C. Anderson, M. Beverridge, C. A. Callahan, L. Justice. **Mastering Windows Server 2003,** Sybex Pub., April 2003.

[1] Event

٤-٥- دستورکار

توجه:
اگر در آزمایش خواسته شده است که خروجی را به مربی نشان دهید، حتما پس از انجام آن، خروجی را به مربی نشان دهید تا برای شما نمره‌ی مربوطه را ثبت نماید.

۱. به Start سپس Run بروید و دستور perfmon را نوشته و کلید Enter را بزنید.
الف)با کلیک راست بر فضای خالی نمودار سمت راست از پنجره‌ی Add Counters منوی پایین افتادنی[1] Performance Object، سه مورد از شیءهایی (Performance Objects) را که برای شما آشناست یادداشت نمایید.
ب) برای شیء Processor، شمارنده Processor Time (Counter) را انتخاب نمایید و کلید Explain را بزنید و بر اساس شرح داده شده، بنویسید که این شمارنده، چه اطلاعاتی درباره‌ی CPU در اختیار قرار می‌دهد. نحوه‌ی محاسبه‌ی زمان برای این شمارنده را نیز که در شرح ذکر شده است، بنویسید.
ج) بر اساس آنچه در سیستم عامل خوانده‌اید یا آنچه در بخش Explain مربوط به شمارنده‌های زیر می‌بینید، تفاوت آن دو را ذکر نمایید:

- Privileged Time
- User Time

د) حال از همان شمارنده‌های پردازنده، Interrupts/sec که بیانگر تعداد وقفه‌های سخت‌افزاری سیستم‌عامل است را انتخاب کنید و کلید Add را بزنید تا به نمودارهای System Monitor اضافه شود و سپس تعداد متوسط وقفه‌های سخت‌افزاری سیستم خود را بر اساس خروجی پنجره‌ی System Monitor، در برگه‌ی گزارش کار یادداشت نمایید.

۲. الف) نمودار تعداد نخ‌های (Threads) فرایند explorer را از شمارنده‌ی Process بیابید، مقدار این نمودار چقدر است؟ حال چند پوشه‌ی مجزا در درایوهای مختلف باز کنید و مقدار جدید شمارنده را بنویسید. برداشت خود را از این فرایند بنویسید. آیا به ازای هر پنجره‌ی باز شده در سیستم‌عامل ویندوز یک فرایند جدا اختصاص یافته است؟ چرا؟ این امر چه سودی دارد؟

[1] Drop Down

ب) چند نمونه (instance) فرایند در سیستم وجود دارد. تعداد را بنویسید.
ج)نمودار زمان بی‌کاری هارد را به دست آورید ؟ یک برنامه‌ی سنگین باز کنید و تغییرات نمودار شمارنده را مشاهده کنید و تفسیر خود را از این تغییرات بنویسید.

۳. به نظر شما کاربرد شمایل چراغ  در نوار ابزار در System Monitor چیست؟

راهنمایی: پیشنهاد می‌شود یک یا دو شمارنده‌ی دلخواه را به System Monitor اضافه نمایید و بعد از شمایل چراغ استفاده نمایید.

۴. با رفتن به Counter Log از Performance Logs and Alerts و انتخاب Log ایجاد شده‌ی System Overview، سوالات زیر را در مورد آن پاسخ دهید:
الف) نام کنونی و محل ذخیره این فایل Log کجاست؟
ب) این Log File از چه اشیایی و چه شمارنده‌هایی از آن‌ها گرفته می‌شود؟
ج) بازه‌ی زمانی ثبت مقادیر شمارنده‌ها در این Log چقدر است؟
د) نوع کنونی این Log File، دودویی (Binary) است، با رفتن به سربرگ Log Files، نوع آن را به متنی که عناصر آن با کاما جدا شده‌اند تغییر دهید. همچنین بنویسید چگونه می‌توان کاری کرد که اندازه‌ی آن بیش از ۲ مگابایت نشود. نیز در کدام گزینه می‌توان تغییراتی ایجاد کرد که در انتهای نام فایل، تاریخ میلادی کنونی ایجاد آن نیز درج شود.
ه) با رفتن به سربرگ Schedule، چگونه می‌توان تغییراتی ایجاد نمود که بتوان شروع ثبت داده‌ها (Log) را به صورت دستی انجام داد، اما پس از یک دقیقه متوقف شود.

توجه: System Overview به دلیل اینکه به صورت پیش‌فرض در سیستم وجود دارد، قابل تغییر نیست و شما باید برای ایجاد یک Log مشابه، تمرین ۵ را انجام دهید.

۵. یک Counter Log با نام دلخواه بسازید که تعدادبایت‌های تبادلی کارت شبکه را هر ۱۰ ثانیه یک‌بار در یک فایل متنی که محتویات آن با کاما از هم جدا شده است، ذخیره کند و انتهای نام فایل به شکل روز/ماه/سال باشد و شروع Logگیری از اکنون تا ۱ دقیقه بعد باشد. فایل خروجی ایجاد شده را به مربی نشان دهید.

۶. یک Trace Log با نام دلخواه بسازید که ورود و خروج‌های دیسک را در یک فایل ترتیبی که حداکثر اندازه‌ی آن یک مگابایت می‌تواند باشد بریزد و از اکنون تا ۳ دقیقه‌ی دیگر شمردن را ادامه دهد.

توجه: در صورت بروز خطا در اجرای این Trace Log، صورت خطا را به دقت بخوانید و راه‌حل آن را بیابید.

راهنمایی: در صورتی که موفق به حل مشکل نشدید، می‌توانید همانگونه که در صورت خطا ذکر شده است و براساس آزمایش دوم در آزمایشگاه سیستم عامل، به Event Viewer مراجعه نمایید و علت دقیق را یافته و مشکل را رفع نمایید.

الف) در هر صورت، راه حل آن را بنویسید.

ب) آیا می‌توانید محتوی فایل Log ایجاد شده را باز کنید و به مربی نشان دهید؟ محتوی این فایل چیست؟

۷. تنظیماتی در alert انجام دهید که هر ۳ ثانیه یک‌بار، بار CPU را اندازه بگیرد و زمانی که بارکاری آن از ۵۰ درصد بیشتر شد، آن را در Event Log ثبت نماید و این امر تا زمانی که به صورت دستی از آن جلوگیری نشده متوقف نشود. خروجی ایجاد شده را به مربی نشان دهید.

آزمایش چهارم

Shell Scripting

۵-۱- مقدمه

تمام سیستم‌عامل‌ها امکاناتی برای کاربران حرفه‌ای فراهم می‌کنند که بتوانند به صورت مستقیم (از طریق خط فرمان) به آن‌ها دستوراتی بدهند. به عنوان مثال دستوری مانند dir در سیستم عامل ویندوز که در حالت خط فرمان اجرا می‌شود، این امکان را به کاربر می‌دهد تا لیست فایل‌های موجود در آدرس جاری را همراه با برخی خصوصیات آن‌ها، رویت نماید. برای تسهیل استفاده‌ی مکرر و چندگانه از دستورات خط فرمان، نوعی از برنامه‌نویسی نیز در این رابطه در اختیار نهاده شده است که به Shell Scripting یا برنامه‌نویسی پوسته‌ی سیستم‌عامل معروف است.

آشنایی با چنین زبانی، علاوه بر ایجاد درک بهتر از یکی از ویژگی‌های مهم سیستم عامل، سودمندی‌های عمده‌ای در محیط کار نیز به همراه خواهد داشت. در این آزمایش، به صورت سریع و مختصر، زمینه‌های آشنایی و استفاده از Shell Scripting فراهم خواهد شد.

توجه: کوییز مربوط به این جلسه نوشتن یک برنامه کوچک با استفاده از Shell Scripting در ویندوز است که دستورات آن در سطح مطالب ذکر شده در پیش آگاهی است، اما ممکن است به صورت ترکیبی استفاده شوند!

۵-۲- هدف

آشنایی با Shell Scripting، دستورات رایج و نکات برنامه‌نویسی آن

۵-۳- پیش‌آگاهی

Shell Script به مجموعه دستورات خط فرمان یا برنامه‌های کوچکی در سیستم‌عامل اطلاق می‌شود که برای تسهیل برخی عملیات مرتبط با سیستم‌عامل، در اختیار کاربر قرار داده شده است. این موضوع به سیستم عامل ویندوز محدود نمی‌شود و در سایر سیستم‌عامل‌ها از جمله لینوکس نیز مورد استفاده است.

Shell Script از زبان‌های اسکریپت به شمار می‌آید، Script بدین معنی است که تعدادی دستورات ساده و محدود در اختیار قرار می‌دهد و Shell یعنی امکان دستور مستقیم از خط فرمان (پوسته) به سیستم عامل فراهم باشد.

غیر از دستورات ساده‌ی خط فرمان، با Windows Shell Scripting می‌توان فایل‌های متنی ساده را ایجاد کرد که‌تحت پسوند bat. یا cmd. ذخیره بشود و توسط خط فرمان ویندوز اجرا گردد. اغلب فایلهای Windows Shell Script در زمان‌کوتاهی ایجاد و تست می‌شوند. به‌طوریکه بیشتر اسکریپت‌های خوب Shell Script بیشتر از ۱۰ تا ۱۵ خط نیستند.

برخلاف بیشتر زبانهای برنامه‌نویسی مدرن، Windows Shell Script شی‌گرا نیست. این سبب می‌شود احتیاجی به یاد گرفتن و اداره مجموعه‌ای ازمحیط‌های توسعه‌یافته نباشد و باعث خواهد شد که فرد به مبانی طراحی برنامه متمرکز شود.

Windows Shell Script قادر است مجموعه فعالیت های زیر را خودکار سازد:

- **فعالیت‌های پیچیده**

شامل هر فعالیتی که هنگام انجام آن به صورت دستی ممکن است خطاهای بسیاری اتفاق بیافتد، مثلاً مدیریت منابع سیستم مثل دیسک درایوها و چاپگرها، مدیریت سرویس‌های ویندوز، پوشه‌ها و درایوهای اشتراکی

- **فعالیت‌های تکراری**

شامل تمام فعالیت‌هایی که قرار است بارها و بارها انجام شوند، مثل پاک کردن فایل‌هایی با پسوند مشخص از پوشه‌ای ویژه.

- **فعالیت‌های حجیم و طولانی**

مشتمل بر فعالیت‌هایی که انجام آنها به صورت دستی بسیار زمانبر است، به طور مثال درست کردن چند صد حساب کاربری جدید.

- **فعالیت‌های زمانبندی شده**

شامل فعالیت‌هایی که در زمان خاصی باید انجام شوند، مثلاً زمانی که مدیران و کاربران از کامپیوترهای خود استفاده نمی‌کنند برنامه‌ای سودمند مانند Disk Defragmenter اجرا گردد.

بدین ترتیب با Windows Shell Script می‌توان به بسیاری منابع ویندوز دسترسی داشت و آنها را مدیریت کرد.

۵-۳-۱- موارد لازم برای شروع کار

اولین چیزی که برای شروع‌کار لازم است سیستم عامل ویندوزی است که Windows Shell Script را پشتیبانی می‌کند که می‌تواند یکی از ویرایش‌های زیر باشد:

Windows NT 4.0 ،Windows2000 ،Windows XP ،Windows2003, Windows Vista, Windows 7

البته مفهوم و امکانات Shell در نسخه‌های اخیر ویندوز دستخوش تغییرات عمده‌ای شده و برتوانایی آن افزوده شده است (Windows Power Shell). همچنین به ویرایشگری که از ایجاد فایل‌های ساده متنی پشتیبانی کند نیز احتیاج داریم. مانند:

Windows Notepad Text Editor

اسکریپت‌ها این امکان را فراهم می‌کنند که برنامه‌های کوچک با صرف وقت و دانش‌کم، مجموعه‌ای از فعالیت‌ها را خودکار کنند.

Windows Shell Script راهی را برای انجام فعالیت‌ها در ویندوز فراهم می‌کند. بدون اینکه نیاز باشد برای انجام آن‌ها از مجموعه‌ای از منوهای گرافیکی ویندوز عبور کنیم. اسکریپت اشتباه‌های تایپی را که درهنگام انجام فعالیت‌ها به‌صورت دستی رخ می‌دهد حذف می‌کند. Windows Shell Script دسترسی کامل به پنجره فرمان ویندوز دارد. Windows Shell فرمان‌های کاربر را تأیید می‌کند و آن را به فرمی که قابل پردازش برای سیستم عامل باشد ترجمه می‌کند و سپس خروجی را که توسط سیستم‌عامل ایجاد شده است در خط فرمان ویندوز نمایش می‌دهد. همچنین دستورات در Shell Script در سیستم‌عامل ویندوز به حروف کوچک و بزرگ حساس نیستند.

کاربران فرمان را در خط فرمان ویندوز می‌نویسند. برای برقراری ارتباط با Windows Shell باید پنجره فرمان ویندوز (CMD) را باز کنیم.

۵-۳-۲- برنامه‌نویسی Shell Script

همانطور که می‌دانید هر زبان برنامه‌نویسی مشتمل بر سه نوع دستور اصلی است:

۱- ترتیب (Sequence)

۲- شرط (Condition)

۳- تکرار (Iteration)

یعنی برای آشنایی سریع و مفید با Shell Script، کافیست که دستورات اصلی آن را در سه زمینه‌ی فوق فراگرفت. در این نوشتار عمده‌ی دستورات از طریق آزمایش و در بخش دستورکار آموزش داده خواهد شد.

تعریف متغیر:

نحوه‌ی تعریف متغیر رشته‌ای[1] به صورت زیر است:

```
Set name=value
```

به عنوان مثال یک نمونه‌ی آن در متن برنامه می‌تواند به شکل زیر باشد:

```
Set msg1= Hello
```

اگر بخواهیم محتوی چنین متغییری را چاپ نماییم، از دستور زیر استفاده می‌کنیم:

```
Echo %msg1%
```

و اگر بخواهیم خروجی را به یک فایل متنی با نام test.txt در همان درایو جاری منتقل نماییم، از عملگر > استفاده می‌کنیم. مانند زیر:

```
Echo %msg1% > test.txt
```

همانطور که می‌دانید، پس از هر دستور Shell و به محض زدن کلید Enter، خط فرمان ویندوز، واکنش نشان داده و خروجی را ارائه خواهد داد یا پیغام خطا و عدم معتبر بودن چنین دستوری را نمایش می‌دهد. برای اجتناب از محاوره‌ای بودن خط فرمان و برای اینکه بتوان چند خط کد را بدون واکنش فوری پوسته‌ی سیستم‌عامل، پشت سر هم نوشت از دستور @echo off استفاده می‌شود. می‌توان با دستور @echo on، وضعیت را به حالت قبل برگرداند.

به عنوان مثال، برنامه‌ی زیر، پس از غیرفعال کردن حالت محاوره‌ای پوسته‌ی سیستم‌عامل، دو متغییر msg1 و msg2 را تعریف خواهد نمود و در خروجی نمایش خواهد داد.

```
@echo off
Set msg1= Hello
```

[1] String

```
Set msg2= There!
Echo %msg1% %msg2%
```

توجه: حتما پس از هر دستور کلیدی مانند set یا echo یا off باید یک فاصله وجود داشته باشد وگرنه تشخیص آن برای سیستم‌عامل، به عنوان یک دستور مجزا امکان‌پذیر نیست و پیامی مشابه به زیر رویت خواهد شد:

'.....' is not recognized as an internal or external command, operable program or batch file.

۵-۳-۳- ایجاد فایل‌های دسته‌ای[1]

با افزوده شدن تعداد دستورات و همچنین بنا به برخی دلایل دیگر که در ادامه ذکر خواهد شد، نیاز است که برنامه‌هایی مانند مثال قبل را در یک فایل بریزیم و با فراخوانی فایل از خط فرمان، آن را اجرا کنیم.

همانطور که یک برنامه به زبان C دارای پسوند c. است و یک برنامه به زبان C++ دارای پسوند cpp. است، یک برنامه به زبان Windows Shell Script نیز دارای پسوند bat. است.

این برنامه‌ها به دلیل آنکه دارای چند دستور خط فرمان به صورت پشت سر هم هستند، به فایل‌های دسته‌ای یا Batch Files معروفند. برای ایجاد آن‌ها کافی است متنی را که در یک ویرایش‌گر متنی ایجاد کرده‌اید با پسوند bat. ذخیره نمایید و در خط فرمان نیز با نوشتن نام کامل و پسوند، آن را فراخوانی کنید.

از دلایل دیگر روی‌آوردن به چنین برنامه‌هایی این است که می‌توان به آن‌ها پارامتر ارسال نمود، به عنوان مثال دستور md که مخفف make directory است، با یک آرگومان که همان نام Directory مطلوب برای ایجاد است، دنبال خواهد شد. مانند دستور زیر که پوشه‌ای به اسم first ایجاد خواهد کرد:

```
Md first
```

در چنین حالتی، md آرگومان ۰ است و first آرگومان ۱ و اگر پس از آن هم با یک فاصله،کلمه‌ی دیگری می‌آمد، آرگومان ۲ می‌شد.

اگر بخواهیم در داخل برنامه‌ی Shell Script از این آرگومان‌ها استفاده کنیم و آن‌ها را از دستور بگیریم، با استفاده از دو علامت % که شماره‌ی آرگومان را احاطه کرده‌اند، این کار را انجام می‌دهیم، مثلاً

echo %1% به معنی چاپ آرگومان دوم خط فرمان است.

[1] Batch Files

۵-٤- مراجع

1. J. L. Ford, **Microsoft Windows Shell Script Programming for the Absolute Beginner**, Premier Press, Nov 2003.

۵-۵- دستور کار

نکته: در املای دستورات بسیار دقت کنید. زیرا بسیاری از خطاها مربوط به عدم وجود فاصله بین دستورات و کلمات کلیدی است.

توجه:

خروجی عملیات خود را در درایو D بریزید و در نهایت پس از آزمایش، تمام فایل‌های مرتبط را حذف نمایید.

۱. به مسیر زیر بروید:

Start -> Run

سپس cmd را نوشته و کلید Enter را بزنید. با تایپ :d و زدن کلید Enter مسیر جاری را به درایو D تغییر دهید. دستور Help را بزنید تا لیست دستورات خط فرمان را ببینید.

الف) سه مورد از دستورات آشنا را به شرح حداکثر یک سطری برای هر یک بنویسید.

ب) دستور زیر چه کاری انجام می‌دهد؟

Dir > test.txt

برای آگاهی از خروجی آن، درایو d را جهت رویت فایل اضافه شده‌ی test.txt بررسی کنید. محتوی فایل چیست؟ اگر مجدداً همین دستور را اجرا کنیم، آیا محتوی فایل test.txt دو برابر می‌شود؟

ج) دستور زیر را وارد نمایید و تفاوت عمگر < با << را بنویسید:

Tree >> test.txt

۲. خروجی برنامه‌ی زیر را بنویسید:

```
@echo off
Set msg1=one
Set msg2=%msg1% two
Set msg3= %msg2% three
Echo  %msg3%
```

۳. خروجی برنامه‌ی زیر چه خواهد بود؟ عملکرد دستورات خط دو و سوم را تشریح کنید.

```
@echo off
Echo set msg=Hello World! > hello.bat
Echo echo %msg% >> Hello.bat
```

۴. همانطور که در پیش‌آگاهی آمد، شماره‌گذاری آرگومان‌های خط فرمان به ترتیب زیر هستند:
%0 %1 %2 %3 %4 %5 %6

همچنین یک **قالب عمومی دستورات شرطی** به شکل زیر است:

```
If [not] string1= =string2 command
```

دستورات زیر را در یک فایل bat با نام test.bat بنویسید و آن فایل را از خط فرمان فراخوانی کنید:

```
cls
@echo off
If  “%1”== “1” echo First Number
If “%1”== “2” echo Second Number
If “%1”== “3” echo Third Number
If “%1”== “” echo no Number
```

نحوه‌ی فراخوانی چگونه باید باشد تا خروجی First Number یا Second Number شود؟ برای یک آرگومان مانند ۳۵۵ خروجی چیست؟

۵. **قالب عمومی دوم دستورات شرطی** به صورت زیر است

```
If [not] exist filename command
```

دستورات زیر را در یک فایل bat با نام test.bat بنویسید و آن فایل را از خط فرمان فراخوانی کنید، پیش از فراخوانی، یک فایل متنی به نام text.txt ایجاد نمایید و پس از فراخوانی بگویید که چه اتفاقی برای text.txt می‌افتد؟

```
@echo off
If exist text.txt (
        Echo removing text.txt
        Del text.txt
        Echo text.txt Removed
)
```

۶. دستور حلقه دارای دو قالب عمومی اول است:

```
For %%variable in (set) do command
```

و

```
For /L %%variable in (start step end) do command
```

برنامه‌ای بنویسید که با استفاده از دستورات حلقه‌ی فوق، اعداد زوج بین ۰ تا ۱۰۰ را چاپ نماید و خروجی را به مربی نشان دهید.

۷. چهار فایل متنی هر کدام با محتوی یک سطر مطالب دلخواه ایجاد کنید که یکی از آن‌ها حتماً با نام bigtxt.txt باشد. سپس برنامه‌ی زیر را که بر اساس قالب عمومی دوم دستور حلقه است نوشته و اجرا کنید و خروجی آن را تفسیر نمایید.

```
@echo off
if exist bigtxt.txt rename bigtxt.txt bigtxt
for %%f in (*.txt) do type %%f >> bigtxt
rename bigtxt bigtxt.txt
```

۸. نوع دوم دستور حلقه می‌تواند به شکل زیر باشد که باید در قسمت بدنه، شرطی برای خروج با استفاده از برچسب[1] END: تعبیه نمود.

:LOOP

بدنه

GOTO LOOP
:END

بر این اساس حدس می‌زنید خروجی برنامه زیر چیست؟ و اگر مایل باشیم حلقه فقط ۵ بار ادامه یابد، چگونه باد آن را فراخوانی کنیم؟

راهنمایی: تازمانی‌که فراخوانی آن را به درستی انجام ندهید، برنامه ممکن است در حلقه‌ی بی‌پایان بیفتد.

```
rem A Bang Counter
@echo off
set n=%1
set i=
: loop
set i=%i%!
        echo Bang!
        if %i%= =%n % goto end
goto loop
:end
```

[1] Lable

آزمایش پنجم

رجیستری (Registry)

٦-١- مقدمه

در درس این جلسه با مفاهیمی آشنا خواهید شد که در عمق سیستم‌عامل ویندوز جای گرفته است. این مفاهیم فاصله شما را با یک کاربر ماهر Windows به علت انتقال آگاهی‌های مفید از ساختار درونی ویندوز، کم می‌کند. Registry به منظورهای مختلفی می‌تواند مورد استفاده قرار گیرد، از نفوذگری[1] گرفته تا تنظیم دقیق[2] و بهینه‌ی سیستم یا رفع‌عیب[3] آن.

شما علاوه بر اینکه نسبت به Registry و نحوه کار با آن آگاهی خواهید یافت، از چگونگی حفظ آن (پشتیبان‌گیری[4]) و برخی نکات مفید دیگر نیز مطلع خواهید شد.

٦-٢- هدف

آشنایی با رجیستری، اهمیت، نحوه کار با آن و نحوه پشتیبان‌گیری[1] از آن

---

[1] Hacking
[2] Tuning
[3] Troubleshooting
[4] Backup

٦-۳- پیش‌آگاهی

در ویندوزهای قدیمی (Windows 3.0) اطلاعات پیکربندی سیستم در فایلی به نام Win.ini نگه‌داری می‌شد، تمام نرم‌افزارها اطلاعاتشان را در این فایل ذخیره می‌کردند که باعث آشفتگی این فایل می‌شد (به‌خصوص از آن نظر که اطلاعاتی که با هم سنخیتی نداشتند، کنار هم نگه‌داری می‌شدند). این آشفتگی، کار با فایل‌ها را مشکل می‌کرد. بعدها ویندوز به یک بانک‌داده‌ای مجهز شد تا هر برنامه همراه با فایل‌های داده‌های مخصوص به خود، اطلاعاتش در یک پایگاه‌داده‌ی جامع نگه‌داری شود.

٦-۳-۱- رجیستری چیست؟

وجود یک پایگاه‌داده‌ی متمرکز که اطلاعات تمام نرم‌افزارها و سخت‌افزارها و کاربران سیستم در آن نگه‌داری شود، علاوه بر این‌که سبب می‌شود *ویندوز* ***ارتباط بین برنامه‌ها را به خوبی برقرار کند*** (زمانی که یک برنامه با برنامه دیگری ارتباط برقرار می‌کند، لازم است تا آدرس و مشخصات دیگر آن برنامه و اجزایش را بداند)، سبب می‌شود تا در ***هنگام باز کردن یک فایل توسط ویندوز*** به سادگی تشخیص داده‌شود که *باید* ***از چه برنامه‌ای استفاده شود***.

Registry یک پایگاه‌داده‌ی سلسله مراتبی است که اطلاعات پیکربندی محیط ویندوز را در خود نگه‌داری می‌کند. Registry از شبکه حمایت می‌کند و اجازه می‌دهد که هر کاربر تنظیمات خود را ذخیره نماید. برنامه‌های کاربردی می‌توانند برای هر کاربر رابط کاربر سفارشی[2] داشته باشند. داده‌های آن‌ها در یک محل ذخیره می‌شود و به بانک داده‌ای ایستگاه کاری که کاربر از آن‌جا به سیستم وارد می‌شود، تزریق می‌شود و این سبب می‌گردد، هر گاه کاربر از هر سیستم در شبکه Log in نماید، همان Desktop سابق خود را ببیند (البته این امکان نیز وجود دارد که به ازای هر سیستم یک Desktop جدا داشته باشد).

اعمال مربوط به shell نیز در Registry ذخیره می‌شوند (در هنگام کار با windows[3] زمانی که کاربر دکمه راست ماوس را کلیک می‌کند، یک منوی شناور[4] ظاهر می‌شود که در آن لیست اعمال برنامه‌ریزی شده از قبیل open، Edit، Rename و ... وجود دارد، به این اعمال، اعمال پوسته‌ای می‌گویند). کاربر با استفاده از Registry می‌تواند اعمال جدیدی تعریف کند و یا اعمال قبلی را تغییر دهد.

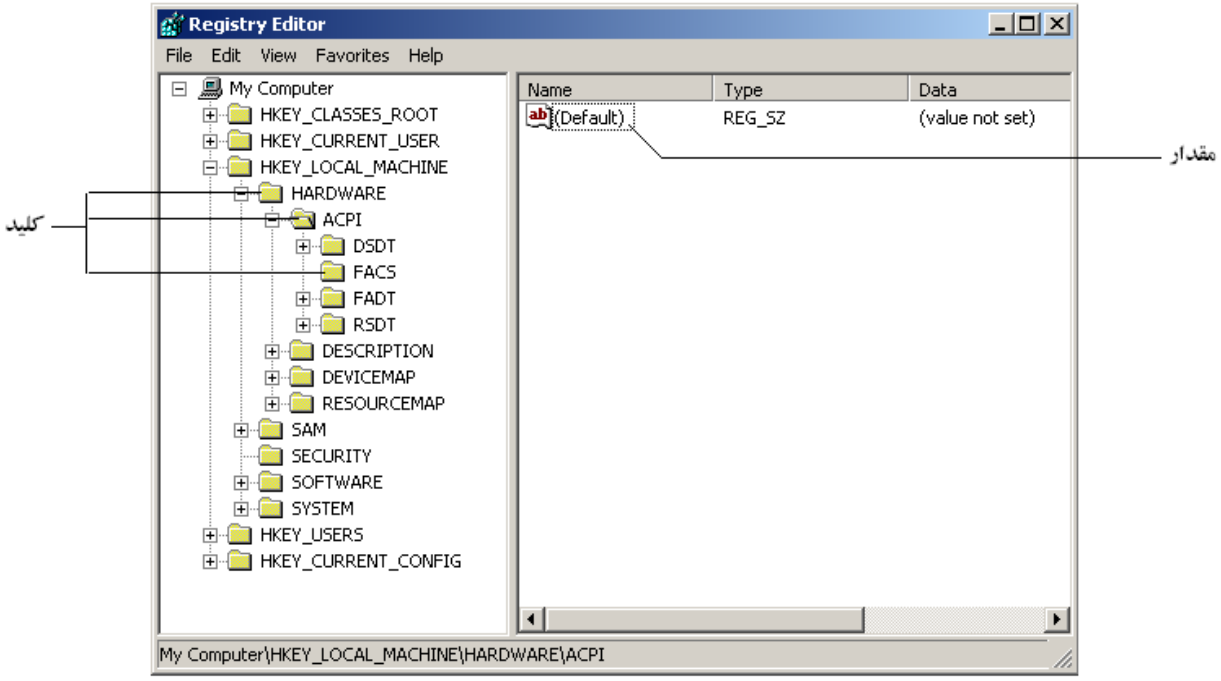
---

[1] Backup
[2] Customized User Interface
[3] یا به زبان بهتر windows Explorer
[4] Pop_up Menu

٦-۳-۲- معماری Registry

یک راه برای آشنایی با Registry استفاده از برنامه Regedit است. این برنامه به کاربران اجازه‌ی رویت و تغییر Registry، همچنین ارتباط با Registryهای دیگر کامپیوترها را می‌دهد.

قالب نمایشی Registry در برنامه Regedit بسیار شبیه به آن‌چه است که Windows برای نمایش شاخه‌ها (Folders) و فایل‌ها (Files) به کار برده است؛ زیرا ساختار درختی دارد. Registry به آن‌چه در ویندوز پوشه یا شاخه نامیده می‌شود، کلید (Key) و به آن‌چه که در ویندوز فایل نامیده می‌شود، مقدار (Value) می‌گوید. شکل زیر مفاهیم را به صورت تصویری بیان می‌کند.



شکل ٦-۱ اجزای رجیستری

هر کلید می‌تواند شامل چند زیرکلید و چند مقدار باشد. مقدار (Value) نیز شامل سه جزء است: نوع داده که به صورت شمایل (Icon) ظاهر می‌شود، نام مقدار و خود مقدار (ارزش مقدار).

برای مقدار دو نوع داده وجود دارد:

- نوع دودویی:

بیشتر اطلاعات عناصر سخت‌افزاری به شکل دودویی ذخیره می‌شوند و در Regedit می‌توانند به فرم دودویی یا شانزده‌شانزدهی (Hexadecimal) به نمایش درآیند. معمولاً تنها کاربران حرفه‌ای

با این مقادیر کار می‌کنند (تغییر این مقادیر نیازمند آشنایی کامل با مشخصات عنصر مربوطه است).

- نوع متنی

برخی اطلاعات به فرم رشته‌ای از کاراکترها به نمایش درمی‌آیند. معمولاً کاربر این نوع داده را تغییر می‌دهد مسیر برنامه‌های کاربردی یا اجزای آن‌ها نام فایل‌ها و ... از این دسته‌اند.

اندازه رجیستری از لحاظ منطقی نامحدود است و می‌تواند هر اندازه بزرگ باشد، اما طبعاً توسط اندازه فیزکی دیسک سخت محدود خواهد شد.

٦-٣-٣- ساختار Registry در ویندوز

ساختار Registry در Windowsهای مختلف، مشابه است، این بدان معنی است که برنامه‌هایی که برای ویندوز نوشته شده‌اند، می‌توانند انتظار داشته باشند، داده‌ها را محل مشترکی بیابند (مستقل از نوع داده) و در کار شبکه نیز سبب تسهیل ارتباط سیستم‌عامل‌های مختلف با هم می‌شود (مثلاً Windows 2000 Advanced Server ، Windows XP Professional و ...).

٦-٣-٤- کاربرد کلیدهای ریشه‌ای در Registry

این کلیدها که با تفکیک از هم، نظامی بهتر به داده‌های Registry بخشیده‌اند در زیر به‌طور خلاصه تشریح شده‌اند:

HKEY_LOCAL_MACHINE

اغلب با عنوان HKLM یاد می‌شود و شامل اطلاعاتی در مورد سخت‌افزار نصب شده، تنظیمات نرم‌افزاری، پروتکل‌های شبکه و ... است.

HKEY_CLASSES_ROOT

اغلب با HKCR نام برده می‌شود و در واقع اشاره‌گری است به شاخه‌ای دیگر در HKEY_LOCAL_MACHINE که اطلاعات تنظیمات نرم‌افزاری را در بر دارد. این داده‌ها

شامل تعاریف و نوع سندها، وابستگی پسوندها و فایل‌های اجرایی، واسط پوسته[1]، میانبرها[2]، چارچوب واسط کاربر ویندوز و ... هستند.

HKEY_USERS

اغلب با عنوان HKU یاد می‌شود و به اطلاعات تمام کاربرانی که به سیستم وارد شده‌اند (Log on)، اشاره دارد. کاربرانی که به سیستم وارد شده‌اند می‌توانند به اطلاعات عام و اطلاعات خاص خودشان دسترسی پیدا کنند. این اطلاعات شامل تنظیمات پیش‌فرض برنامه‌های کاربردی، پیکربندی میزکار[3] و مانند این‌هاست.

HKEY_ CURRENT_CONFIG

اغلب با HKCC یاد می‌شود و اطلاعات پیکربندی کنونی سخت‌افزار کامپیوتر در این‌جا نگه‌داری می‌گردد.

HKEY_CURRENT_USER

با HKCU نیز نام برده می‌شود و زیرکلیدی است از HKey_Users که به شاخه‌ی مربوط به کاربر فعلی اشاره می‌کند.

[1] Shell Interface
[2] Shortcut
[3] Desktop

٦-٤- دستورکار

توجه:

- با توجه به اینکه تغییرات در Registry، راه‌برگشت (Undo) ندارد، لذا *حتماً در حیطه آزمایش‌ها جلو بروید*. هر گونه تغییر نابجا ممکن است موجب از دست رفتن ویندوز شود.

۱. یک راه پشتیبان‌گیری[1] از کلیدها و مقادیر در رجیستری آن است که بر کلید مورد نظر کلیک راست نمایید و گزینه‌ی Export را انتخاب نمایید. فرایند زیر را طی نمایید تا دریابید این روش پشتیبان‌گیری، چه تفاوتی با پشتیبان‌گیری‌های معمول دارد؟

ابتدا بر HKEY_CURRENT_USER کلیک راست نمایید، سپس گزینه‌ی New و Key را انتخاب نمایید و نام کلید را test قرار دهید. برکلید Test کلیک راست نمایید و کلید دیگری با نام in1 بسازید.

حال برکلید Test کلیک راست نمایید و گزینه‌ی Export را انتخاب نمایید و محتوی این کلید را با نام Backup در درایو D در پوشه‌ایی با نام Work ذخیره کنید.

اکنون برکلید Test کلیک راست نمایید و کلید دیگری با نام in2 بسازید.

حال کلید in1 را حذف نمایید و Registry را ببندید و باز کنید و از منوی File گزینه‌ی Import را انتخاب کنید و فایل Backup را از درایو D، پوشه‌ی Work انتخاب نمایید تا مقدار Test را بازیابی کنید.

الف) حاصل کار چیست؟ آیا در فرایند بازیابی مقدارپیشین کلید Test، In2 حذف شده است؟

ب) پسوند فایل Backup چیست؟

ج) محتوی فایل Backup را بنویسید (آن را با محیطی مانند Notepad باز کنید)

د) اگر بخواهیم با تغییر در فایل Backup، گزینه‌ایی اضافه کنیم که سبب ایجاد کلید In3 در زیر کلید Test گردد، چه سطری باید با آن اضافه کنیم؟

ه) اگر بخواهید با همین ایده از کل رجیستری پشتیبان بگیرید، از پنجره Registry Editor بر شمایل My Computer کلیک راست نمایید و Export را انتخاب نمایید، و سپس مسیر D:\Work را به آن بدهید، به این ترتیب شما از هر چیز از سراسر Registry، نسخه پشتیبان گرفته‌اید. اما این روش پشتیبان‌گیری ممکن

[1] Backup

است مشکلاتی به همراه داشته باشد، به عنوان مثال در بازیابی، ویندوز اجازه‌ی جایگزینی مقادیر مورد استفاده را نخواهد داد، زیرا رجیستری جزو فایل‌های مورد استفاده سیستم‌عامل به حساب می‌آید. برای اجبار ویندوز به این کار، چه راهی پیشنهاد می‌کنید؟ (از آموخته‌های جلسات پیشین استفاده کنید)
نکته: راه درست و ایمن پشتیبان‌گیری از رجیستری استفاده از نرم‌افزار سودمند Backup تعبیه شده در ویندوز است.

۲. برنامه Regedit را از طریق RUN اجرا کنید. کلید HKEY_CLASSES_ROOT را باز کنید. زیرکلیدهای آن که با نقطه شروع می‌شوند، قالب پرونده‌هایی (File Format) هستند که در سیستم تعریف شده‌اند.
الف) دو مورد آشنا از آن‌ها را بنویسید.
ب) به کمک برنامه Windows Explorer و از طریق گزینه Tools -> Folder Option به سربرگ File Types بروید و در آن‌جا نوع فایل جدیدی تعریف کنید. پسوند آن را ZWQ بنامید (File Extension)، که سیستم به‌طور خودکار آن را نوع فایل‌های ZWQ متناظر[1] خواهد کرد. به برنامه Regedit بازگشته و پس از Refresh نمودن، Zwq را جستجو نمایید (برای تسهیل جستجو تنها کلیدها را جستجو کنید و همیشه توجه داشته باشید جستجو از جایی شروع می‌شود که هم‌اکنون highlight کرده‌اید) پسوند اضافه شده و مقدار متغییر Default آن را بیابید و در برگه بنویسید. در همان ردیف یک کلید با نامی یکسان با مقدار متغییر Default وجود دارد، آن را نیز بیابید.
راهنمایی: به عنوان مثال اگر پس از یافتن zwq. متوجه شدید که داده متغییر Default آن ft000002 است، اکنون باید دنبال کلیدی به همان نام در ادامه جستجو کنید.

۳. در کلیدی که در تمرین قبل یافتید، فعلاً هیچ زیر کلیدی وجود ندارد. در شاخه‌ی D:\Work یک پرونده متنی با نام st.zwq ایجاد نمایید. روی آن کلیک راست کنید. می‌بینید که غیر از اعمال عادی، عمل خاصی روی آن تعریف نشده است. باز هم با کمک Windows Explorer و از طریق Folder Option از منوی Tools به سربرگ File Types بروید. قالب پرونده zwq را انتخاب کنید و بر کلید Advanced و سپس New کلیک کنید و دو عمل (Action) با نام‌های Simple Edit (مرتبط با برنامه Notepad.exe) و Advanced Edit مرتبط با خط فرمان(cmd.exe) برای آن تعریف کنید. بار دیگر روی فایلی که در شاخه

[1] Associate

D:\Work ایجاد کرده‌اید، کلیک راست کنید و گزینه‌هایی که به آن منو اضافه شده است را مشاهده کنید (این منو را Context Menu یا منوی زمینه می‌نامند). می‌بینید که کلید Shell در زیر کلید ft000001 ایجاد شده است. چه کلیدهایی با چه مقادیری در زیر آن ایجاد شده‌اند؟ اگر بخواهیم با تغییر در رجیستری کاری کنیم که منوی Advanced Edit با برنامه‌ی Microsoft Word باز شود، چه تغییری باید در رجیستری اعمال کنیم؟

۴. با دوبار کلیک کردن ماوس روی پرونده st.zwq، پوسته‌ی ویندوز عمل پیش‌فرضی را که در مقدار متغییر default کلید Shell مربوطه تعیین شده است، اجر می‌کند (shell به معنای پوسته است). دو کلیدی که در تمرین سوم اضافه شده بود را از رجیستری حذف کنید و دوباره روی همان فایل دوبار کلیک کنید. چه اتفاقی می‌افتد؟ چرا؟ در پنجره ظاهر شده برنامه Notepad را انتخاب کنید و گزینه پایین پنجره را علامت بزنید. آیا متناظر با چنین تنظیمی، در رجیستری تغییری روی داده است؟ آیا می‌توانید آن را بیابید؟ (در صورتی که تغییری اعمال نشده باشد، پس از یک logoff آن را بررسی کنید).

راهنمایی: اگر باز هم موفق به یافتن تنظیمات اضافه شده به رجیستری نشدید، به آدرس زیر مراجعه کنید و کلید مذکور را به همراه زیر کلیدها و تنظیمات بیابید و ذکر نمایید:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts

۵. با جستجو در رجیستری موارد زیر را بیابید:

الف- چه کاغذ دیواری (WallPaper) برای Desktop انتخاب شده است؟ آدرس جایی از رجیستری که در آن کاغذ دیواری تعیین شده است را بنویسید.

تذکر: می‌توانید برای تغییر کاغذ دیواری مقدار این متغییر را عوض کنید، اما برای آن‌که این تغییر اعمال شود باید کامپیوتر خود را Reset نمایید.

ب- آدرس جایی از رجیستری را که اطلاعات برنامه‌ی Add\Remove Programs در برگه Install\Uninstall در آن قرار دارد بیابید.

۶. در رجیستری کلیدهای زیر را بیابید:

الف- کلید

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU

را بیابید و با مقایسه‌ی آن با لیست پایین افتادنی که در پنجره Run وجود دارد، هدف این کلید را پیدا کنید.
(MRU مخفف Most Recently Used است) معنای مقدار متغییر MRUList چیست؟

ب- در کلید

HKEY_CLASSES_ROOT\CLSID\{20D04FE0-3AEA-1069-A2D8-08002B30309D}\shell

مقدار متغییر default را "find" قرار دهید. با دو بار کلیک روی My Computer چه اتفاقی می‌افتد.

توجه: حتماً پس از انجام این آزمایش، وضعیت را به حالت قبل برگردانید.

۷. الف) پیش‌واکشی (Prefetching) به چه معنی است؟

ب) به آدرس C:\WINDOWS\Prefetch در ویندوز بروید، بر اساس نام کامل فایل‌ها تصور می‌کنید چه فایل‌هایی پیش‌واکشی شده‌اند؟

ج) به آدرس

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters\EnablePrefetcher

در رجیستری بروید و مقدار آن را به ۲ تغییر دهید. پس از خالی کردن محتوی پوشه‌ی C:\WINDOWS\Prefetch سیستم را Restart کنید و محتوی آن را مجدداً رویت نمایید. در این حالت فایل‌های لازم برای بوت شدن ویندوز پیش‌واکشی می‌شوند و ویندوز سریع‌تر شروع به کار خواهد کرد. مقدار تغییر یافته را به ۳ برگردانید.

۸.آیا تمام تغییراتی که در رجیستری می‌دهیم، بلافاصله اثرات آن‌ها منعکس می‌شود؟ چند روش دیگر برای اعمال تغییر می‌توانید نام ببرید؟

اختیاری:

۹. آیا می‌توانید با کمک Registry نام My Computer و Recycle Bin را به ترتیب به "کامپیوتر من" و "سطل آشغال" تبدیل نمایید؟ چگونه؟

آزمایش ششم

سیستم‌فایل در ویندوز

۷-۱- مقدمه

یکی از مهمترین قسمت‌های سیستم‌عامل، سیستم فایل است. سیستم‌فایل[1] مشخص می‌کند که سیستم‌عامل چگونه اطلاعات را روی دیسک ذخیره، پاک می‌کند و جابه‌جا کند یا آن‌را بخواند. به‌عبارت دیگر سیستم فایل، مدیریت فایل‌ها و پوشه[2]های روی هارددیسک را به عهده دارد. از مهمترین و پرکاربردترین سیستم‌فایل‌ها عبارتنداز:

FAT, NTFS, HPFS, Netware Fs, Linux Ext2, Linux Ext3, Linux Swap

برای ذخیره اطلاعات و نصب سیستم‌عامل پیش از هر چیز باید دیسک‌سخت را پارتیشن‌بندی کرد سپس هر پارتیشن را با سیستم‌فایل موردنظر قالب‌بندی[3] نمود. حال این سوال که از میان سیستم‌فایل‌های موجود کدام را برای فرمت‌دهی انتخاب کنیم، اهمیت آشنایی با ویژگی‌های سیستم فایل‌ها نمایان می‌شود.

۷-۲- هدف

هدف کلی از انجام این آزمایش، آشنایی با مفهوم فایل و سیستم‌فایل‌های مربوط به ویندوز است. در طی انجام این آزمایش با ساختار هارددیسک و جدول‌های سیستم‌فایل آشنا خواهید شد. همچنین انتظار می‌رود دانشجو با برخی از اجزای این ساختارها آشنا شده و نتایج را به‌صورت عینی مشاهده نماید.

---

[1] File System or Filesystem
[2] Folder
[3] Format

۷-۳- پیش‌آگاهی

سیستم‌فایل، سیستمی است که با بهره‌گیری از مکانیزمی خاص وضعیت ذخیره‌سازی، یافتن و دسترسی به فایل‌ها را سازماندهی می‌کند. می‌توان گفت این سیستم مسئول ذخیره، نام‌گذاری و بازیابی اطلاعات سیستم‌عامل و کاربر در قالب مفهوم منطقی فایل، بر روی حافظه‌ی فیزیکی است. سیستم‌فایل‌ها در سه دسته‌ی Disk File System، Network File System، Special Purpose File System تقسیم‌بندی می‌شوند که هدف ما در این آزمایش بررسی Disk File System های سیستم‌عامل ویندوز است. در ادامه مفاهیم مربوط به سیستم‌فایل به‌صورت جامعی خواهد آمد.

۷-۳-۱- ساختار دیسک‌سخت و تقسیمات آن

همان‌طور که می‌دانید دیسک‌سخت به عنوان یک قطعه مکانیکی از اجزای اصلی زیر تشکیل شده است:

- ✓ **شیار**[1]**:** محل ذخیره بیت‌های اطلاعات در هر سطح[2] هستند. شیارها معمولا به صورت دوایر متحدالمرکز هستند و تعداد سکتور در شیار از ۴ تا ۳۲ در فلاپی‌ها گاه تا ۱۰۰در انواع دیگر حافظه متغیر است.
- ✓ **استوانه**[3]**:** تمام شیارهای با شعاع یکسان (از رویه‌های مختلف) تشکیل یک استوانه را می‌دهند.
- ✓ **سکتور**[4]**:** تقسیماتی است از شیار با اندازه مساوی. هر شیار از تعدادی سکتور تشکیل شده‌است و اندازه سکتور از ۳۲ بایت تا ۴۰۹۶ بایت متغیر و معمولا ۵۱۲ بایت است.

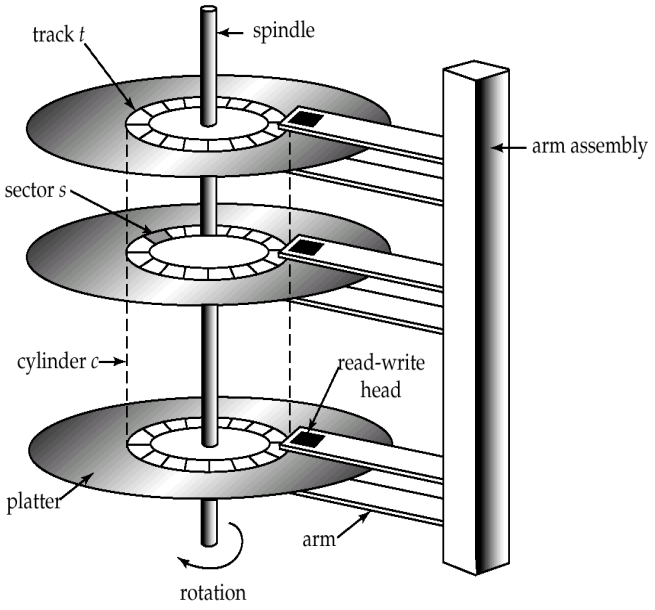
دو نوع سکتور وجود دارد:

- ✓ **سکتور سخت‌افزاری:** که توسط سازنده ایجاد می‌شود (فرمت سطح پایین)
- ✓ **سکتور نرم‌افزاری** (بلاک): از طریق نرم‌افزار (سیستم‌عامل) قابل ایجاد است. و با فرمت کردن نرم‌افزاری ایجاد می‌شود.
- ✓ **خوشه یا کلاستر**[5]**:** یک مفهوم منطقی بوده و اندازه آن توسط سیستم فایل تعیین می‌شود و برای ذخیره‌ی داده‌های موجود در فایل مورد استفاده قرار می‌گیرد. به عبارتی، کلاستر کوچک‌ترین

[1] Track
[2] Platter
[3] Spindle
[4] Sector
[5] Cluster

قطعه با قابلیت ذخیره‌سازی داده بر روی دیسک‌سخت است. درصورتی‌که سکتور یک مفهوم فیزیکی است و از طریق سیستم‌عامل قابل تغییر نیست.

هر کلاستر شامل گروه‌های متوالی از سکتور می‌باشد و معمولاً از ۱ یا ۲ یا ۴ یا ... ۶۴ سکتور تشکیل شده است. هر کلاستر دارای یک آدرس و شماره‌ی اولین کلاستر ۲ است (کلاسترهای ۰ یا ۱ برای مقاصد رزرو کنار گذاشته شده‌اند). تعداد سکتورهای یک کلاسور با کارایی سیستم رابطه غیر خطی دارد و می‌تواند بر سرعت یا حافظه مصرفی موثر باشد. به عنوان مثال اگر فایلی از یک کلاستر هم کوچک‌تر باشد، سیستم ناچار است کل آن کلاستر را به فایل مذکور اختصاص دهد.



شکل ۷-۱ - نمایی از اجزای هارددیسک[1]

۷-۳-۲- فایل

فایل نیز یک مفهوم منطقی است که توسط سیستم‌عامل تولید شده به حافظه‌ی فیزیکی نگاشته می‌شود. به‌عبارت دیگر سیستم‌عامل واحدهای حافظه‌ی فیزیکی را به‌صورت منطقی و توسط فایل انتزاع[1] می‌بخشد لذا می‌توان گفت که فایل یک Abstract Data Type است. اطلاعاتی که در فایل‌ها وجود دارد، توسط ایجاد کننده فایل مشخص می‌گردند. برای مثال ممکن است یک فایل حاوی متن[2]، برنامه‌های اجرایی[3]،

[1] Abstract
[2] Text
[3] Executable programs

عکس‌های گرافیکی[1] و ... باشد که به نوع آن بستگی دارد. در فایل‌سیستم FAT پوشه نیز ساختاری شبیه به فایل دارد با این تفاوت که در فایل یک بیت به علامت زده می‌شود تا از یکدیگر جدا شوند در ادامه بیشتر به آن پرداخته می‌شود.

۷-۳-۳- فایل‌سیستم‌های ویندوز

شرکت مایکروسافت در ویرایش‌های مختلف سیستم‌عامل‌های خود، نسخه‌های مختلفی از فایل‌سیستم را ارائه داده است که برخی از آنها عبارتنداز: FAT12، FAT16، FAT32، exFAT (FAT64) و NTFS.
(لازم به ذکر است که FAT12 در فلاپی درایوها و exFAT برای نسخه‌های جدید ویندوز از قبیل ویندوز۷، ویندوز سرور ۲۰۰۸ و Vista SP1 در فلش مموری‌ها کاربرد دارند).

۷-۳-۳-۱- جدول تخصیص فایل - FAT[2]

سیستم‌فایل FAT، حاوی جدولی شامل وضعیت کلاسترها و اشاره‌گری به کلاستر بعدی می‌باشد. پس در هر مدخل آن شماره‌ی یک کلاستر قرار دارد. در جدول ۱، لیست انواع سیستم‌فایل FAT همراه با برخی ویژگی‌های آن مشاهده می‌شود. برای درک بهتر از چگونگی محاسبه این مقادیر به مثال زیر توجه کنید

جدول FAT12

برای محاسبه مقدیر جدول۱ در Fat12 به موارد زیر نیاز داریم:

- ✓ بیشینه‌ی تعداد کلاستر
- ✓ بیشینه حجم قابل پشتیبانی درایو
- ✓ حجم کلاستر قابل پشتیبانی

بیشینه حجم قابل پشتیبانی درایو = حجم کلاستر قابل پشتیبانی ✕ بیشینه‌ی تعداد کلاستر

لذا داریم

بیشینه حجم قابل پشتیبانی درایو = ۴ ✕ ۴۰۹۶ = ۱۶ مگابایت

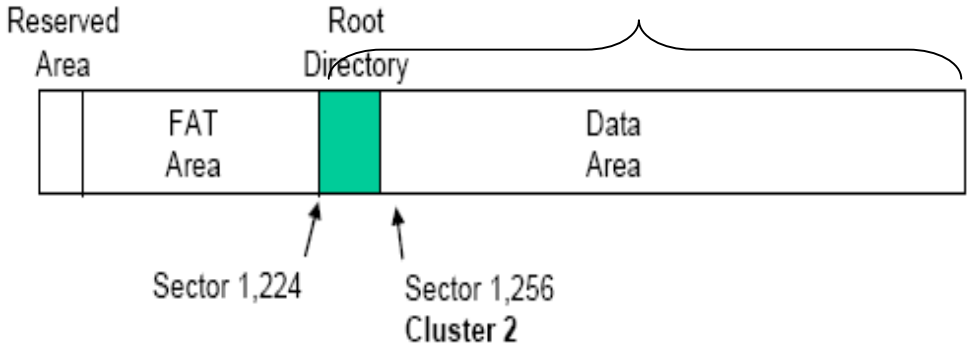
[1] Graphic image
[2] File Allocation Table

جدول ۷-۱- سیستم‌فایل‌ها در ویندوز[3]

فایل‌سیستم	سال ایجاد	اندازه‌ی هر مدخل FAT	بیشینه تعداد کلاسترها (به‌صورت تقریبی)	حجم کلاستر قابل پشتیبانی	بیشینه حجم قابل پشتیبانی درایو (به‌صورت تقریبی)
FAT12	1977	12 bits	4,096	512 B تا 4 KB	16 MB یا 16,736,256 B
FAT16	1987	16 bits	65,536	2 KB تا 32 KB	2 GB یا 2,147,123,200 B
FAT32	1996	28 bits	268,435,456	4 KB تا 32 KB	2 TB یا $B2^{41}$
exFAT	2006,2009	64 bits	4294967295	512 B تا 32 MB	512 TB

در شکل زیر (شکل ۲)، محل قرار گرفتن FAT و همچنین Root Directory را در هارددیسک مشاهده می‌کنید. همانطور که اشاره شد اولین کلاستر، کلاستر ۲ می‌باشد (بعد از Root Directory). البته موقعیت قرارگیری کلاستر ۲ در انواع FATها متفاوت است.

اگر فرض شود اندازه‌ی کلاستر، ۲۰۴۸ بایت (معادل با ۴ سکتور) و اندازه‌ی داده‌ی سکتور اولیه ۱۲۲۴ بایت باشد، اولین سکتور که در Data Area قرار دارد به Root Directory اختصاص داده می‌شود که اندازه Root Directory، ۳۲ بایت است. در نتیجه کلاستر ۲ بعد از Root Directory (سکتور ۱۲۵۶) قرار می‌گیرد: ۱۲۵۶ = ۳۲ + ۱۲۲۴.



شکل ۷-۲- ساختار دیسک سخت[3]

جدول ۷-۲ – بازه هر کاربرد در دیسک سخت[3]

سکتور(ها)	آدرس	کاربرد
0	0x0000-0x01FF	Boot Sector
1-9	0x0200-0x13FF	File Allocation Table (Primary)
10-18	0x1400-0x25FF	File Allocation Table (Secondary)
19-32	0x2600-0x41FF	Root Directory
33-2879	0x4200-0x167FFF	File Storage Space

جزییاتی در مورد Boot Sector و Root Directory

اولین سکتور در سیستم فایل **Boot Sector** می‌باشد که شامل اطلاعاتی در مورد نوع سیستم‌فایل و اندازه و محل قرارگیری ساختار داده‌ی سیستم‌فایل می‌باشد. فرمت Boot Sector در FAT های مختلف متفاوت است. جدول‌های ۳ و ۴ فرمت Boot Sector را در سیستم‌فایل‌های مختلف نشان می‌دهد. برای آشنایی بیشتر و برای درک میزان تطابق مفاهیم نظری، با یافته‌های عملی حاصل از اجرای نرم‌افزار تحلیل فایل،در بخش‌های بعدی، مروری بر جداول زیر سودمند خواهد بود.

جدول ۷-۳ - بایت‌های 0 تا 35 Boot Sector مشترک در سیستم‌فایل‌های FAT12 و FAT16 و FAT32 [3]

بایت‌ها	کاربرد
0-2	Assembly code instructions to jump to boot code (mandatory in bootable partition)
3-10	OEM name in ASCII
11-12	Bytes per sector (512, 1024, 2048, or 4096)
13	Sectors per cluster (Must be a power of 2 and cluster size must be <=32 KB)
14-15	Size of reserved area, in sectors
16	Number of FATs (usually 2)
17-18	Maximum number of files in the root directory (FAT12/16; 0 for FAT32)
19-20	Number of sectors in the file system; if 2 Byte is not large enough, set to 0 and use 4 B value in bytes 32-35 below
21	Media type (0xf0=removable disk, 0xf8=fixed disk)
22-23	Size of each FAT, in sectors, for FAT12/16; 0 for FAT32
24-25	Sectors per track in storage device
26-27	Number of heads in storage device
28-31	Number of sectors before the start partition
32-35	Number of sectors in the file system; this field will be 0 if the 2B field above (bytes 19-20) is non-zero

جدول ۷-٤ - فرمت Boot Sector در FAT12 و FAT16 [3]

بایت‌ها	کاربرد
0-35	جدول ۳
36	BIOS INT 13h (low level disk services) drive number
37	Not used
38	Extended boot signature to validate next three fields (0x29)
39-42	Volume serial number
43-53	Volume label, in ASCII
54-61	File system type level, in ASCII. (Generally "FAT", "FAT12", or "FAT16")

Not used	62-509
Signature value (0xaa55)	510-511

در جدول بالا دو عبارت زیر را می‌بینید که به صورت زیر تعریف می‌شود.

SFN[1]: گونه‌ای از نامگذاری برای فایلهایی است که از محیط ویندوز به Dos آورده شده‌اند.

این فایلها در ویندوز به صورت زیر نامگذاری می‌شوند.

THISIS~1.txt

LFN[2]: از این گونه نامگذاری در ویندوز به صورت پیش‌فرض استفاده می‌شود و به صورت Unicode ذخیره می‌شوند

This is test 1.txt

Root Directory، اسم فایل، طول کلاستر، کلاستر اول و ... را درخود نگهداری می‌کند و بعد از ناحیه FAT قرار دارد. هر مدخل در **Root Directory** قالبی شبیه به زیر دارد:

Root Directory SFN Entry Data Structure

Bytes	Purpose
0	First character of file name (ASCII) *or* allocation status (0x00=unallocated, 0xe5=deleted)
1-10	Characters 2-11 of the file name (ASCII); the "." is implied between bytes 7 and 8
11	File attributes (see File Attributes table)
12	Reserved
13	File creation time (in tenths of seconds)*
14-15	Creation time (hours, minutes, seconds)*
16-17	Creation date*
18-19	Access date*
20-21	High-order 2 bytes of address of first cluster (0 for FAT12/16)*
22-23	Modified time (hours, minutes, seconds)
24-25	Modified date
26-27	Low-order 2 bytes of address of first cluster
28-31	File size (0 for directories)

File Attributes

Flag Value	Description
0000 0001 (0x01)	Read-only
0000 0010 (0x02)	Hidden file
0000 0100 (0x04)	System file
0000 1000 (0x08)	Volume label
0000 1111 (0x0f)	Long file name
0001 0000 (0x10)	Directory
0010 0000 (0x20)	Archive

** Bytes 13-22 are unused by DOS*

شکل ۷-۳ – [3] Root Directory Entry Format

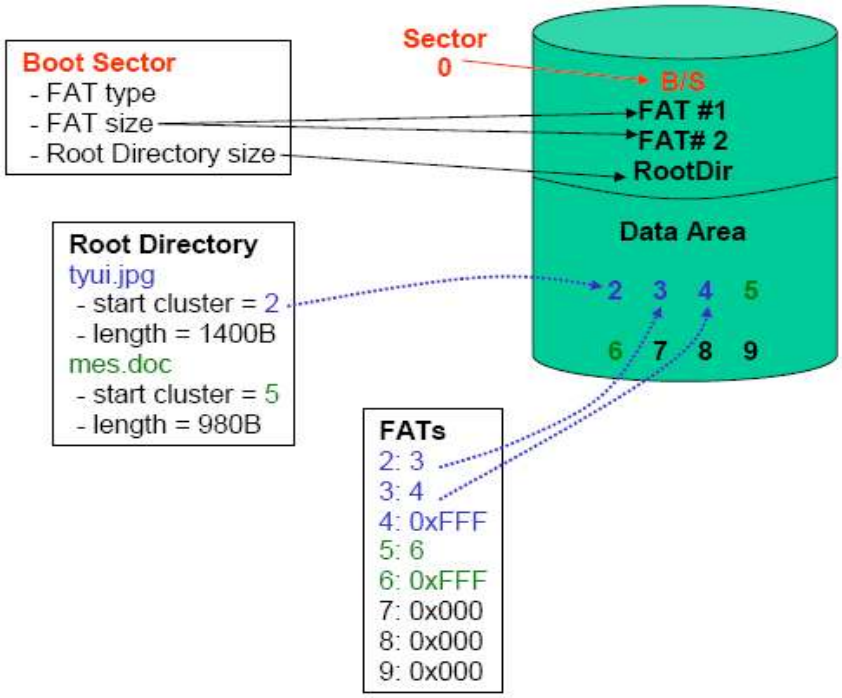
[1] Long File Name
[2] Short File Name

بـرای درک بهتـر مفـاهیم Boot Sector، Root Directory و همچنـین چگـونگی ذخیـره‌ی داده در سیستم‌فایل مثال زیر را در نظر بگیرید:

دو فایل را یکی به نام tyui.jpg باحجم ۱۴۰۰ بایت و دیگری به نام mes.doc با حجم ۹۸۰ بایـت اسـت در نظر بگیرید. اندازه‌ی کلاستر را هم ۵۱۲ بایت فرض شود. همان‌طور که در شکل ۴ نیـز مـشاهده مـی‌شـود، فایل tyui.jpeg متناظر با حجمی کـه دارد سـه کلاسـتر ۲، ۳و ۴ را در اختیـار گرفتـه اسـت و ۱۳۶ بایـت از کلاستر ۴ خالی مانده و هدر می‌رود (زیرا ۱۳۶ + ۳ × ۵۱۲ = ۱۴۰۰). به این فضای هدر رفته، فضای باطلـه[1] گفته می‌شود. به‌طور مشابه، فایل mes.doc نیز کلاسترهای ۵ و ۶ را به خود اختصاص می‌دهد. لازم به ذکر است که سیستم‌عامل، نخستین کلاستر خالی بعدی (اگر پیمایش را به ترتیب فرض کنیم) را برای شـروع و یا ادامه‌ی فایل اختصاص می‌دهد (مشابه الگوی برازش، اولین برازش[2] در تخصیص حافظه).

برای نمونه، به بررسی چگونگی ذخیره‌ی فایل tyui.jpg می‌پردازیم. سیستم‌عامل با استفاده از Root Directory شماره اولین کلاستر فایل، که در این مثال ۲ است، را به عنوان شروع مکان قرار گرفتن فایل در نظر خواهد گرفت و سپس با مراجعه به FAT،"شماره کلاستر عددی که حاوی ادامه‌ی فایل است چیست؟" می‌جوید. FAT پاسخ می‌دهد که شماره کلاستر بعدی، ۳ است سپس این رویه تا جایی ادامه می‌یابد که FAT پاسخ دهد که کلاستر x پایان فایل است. در مدخل مربوط به کلاستر ۳ در FAT، شماره‌ی ۴ درج گردیده و در مدخل کلاستر ۴ مقدار 0xFFF (به انتهای لیست خود اشاره می‌کند)، بدین معنی که کلاستر ۴ پایان فایل tyui.jpg است. این‌گونه است که فایل‌ها در این نوع از سیستم‌فایل سازماندهی می‌شوند.

[1] Slack Space
[2] First Fit



شکل ۷-٤ – نمایی از سیستم‌فایل FAT [3]

به‌طورکلی معایب زیر بر سیستم فایل FAT وارد است:

- سادگی لایه‌ی فایل، که اجازه تکه‌تکه‌شدن[1] آسان آن را به دو صورت Internal Fragmentation و External Fragmentation که منجر به از دست رفتن فضای دیسک سخت می‌شود، می‌دهد.
- FAT خطاهای فیزیکی دیسک سخت را پوشش نمی‌دهد.
- نگارش‌های اولیه FAT (تا پیش از FAT 32)، اجازه‌ی استفاده از نام فایل بیش از ۱۱ کاراکتری را نمی‌دهند (۸ کاراکتر برای نام فایل، ۳ کاراکتر برای پسوند آن). بدین ترتیب مایکروسافت تغییراتی درنگارش‌های بعدی FAT انجام داد و در تکنولوژی به نام VFAT[2] پیاده‌سازی کرد که این امکان را می‌دهد که بتوان ۲۵۵ کاراکتر و بیشتر را به‌عنوان نام فایل قرار داد.
- بر روی کلاسترهای بزرگ، مقدار زیادی از فضای مورد استفاده به دلیل تعدد وجود فایل‌های با ظرفیت کم هدر می‌رود.

[1] Fragmentation
[2] Virtual FAT

۷-۳-۳-۲- فایل‌سیستم فن‌آوری جدید یا NTFS[1]:

NTFS سیستم‌فایل جدیدی است که شرکت مایکروسافت به همراه اولین نسخه Windows NT (از خانواده‌ی ویندوز ۲۰۰۰ به بعد) به بازار عرضه کرد. این سیستم‌فایل، ترکیبی از کارایی، اعتبار و سازگاری و دیگر خصوصیات مطلوبی است که در سایر سیستم فایل‌ها مانند FAT وجود ندارد. طراحی قالب آن طوری است که عملیات استاندارد بر فایل مثل خواندن، نوشتن وجستجو در آن و هر عملیات دیگری، در حوزه‌ی بازیابی آن روی هارد دیسک های با ظر فیت بالا با سر عت انجام می گیرد.

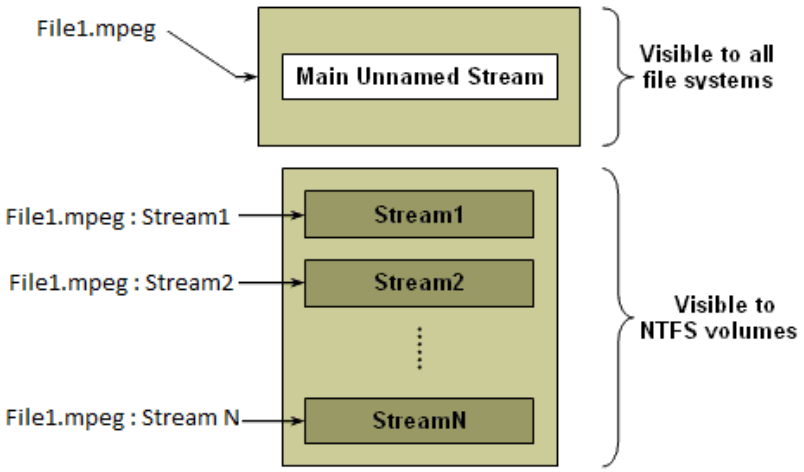
ویژگی‌های سیستم‌فایل NTFS:

- **قابلیت بازیابی[2]:** اگر به هر دلایلی مانند قطع برق، توقف ناگهانی سیستم یا لغو دستور، خواندن و نوشتن اطلاعات دچار مشکل شود، NTFS حفظ سازگاری[3] دیسک را تضمین می کند. یعنی اینکه در آغاز به کار سیستم‌عامل، بلافاصله، NTFS، ناسازگاری‌های احتمالی سیستم‌فایل را بررسی می کند.
- **انعطاف‌پذیری در برابر اشکال در ذخیره اطلاعات:** NTFS به‌طور موثر از روش‌های مضاعف‌سازی[4] اطلاعات برای حفظ اطلاعات حیاتی سیستم‌فایل و همین‌طور نگاشت کلاسترهای معیوب استفاده می کند.
- **امنیت اطلاعات[5]:** در این سیستم‌فایل می‌توان برای فایل‌ها و پوشه‌ها سطح دسترسی مشخص نمود. در این‌صورت سیستم‌عامل از این اطلاعات در برابر دسترسی افراد غیرمجاز جلوگیری می کند که این محدودیت هم برای کاربرانی که از طریق شبکه به اطلاعات دسترسی دارند و چه مستقیماً ازکامپیوتر استفاده می کنند، اعمال می‌شود.
- **رمزنگاری فایل‌ها:** NTFS سیستم‌فایل رمزشده[6] (EFS) را برای حفاظت از اطلاعات با روش‌های رمزنگاری در اختیار قرار می‌دهد. وقتی داده‌ها رمز می‌شود، از لحاظ فیزیکی به‌صورتی است که بدون رمزگشایی بی‌معنی و غیرقابل استفاده است.
- **نامگذاری Unicode**: NTFS از Unicode به‌عنوان مجموعه کاراکترهای استاندارد خود استفاده می کندکه در آن حروف و علائم همه‌ی زبان‌ها وجود دارد.

---

[1] New Technology File System
[2] Data Recoverability
[3] Consistency
[4] Duplicate
[5] Data security
[6] Encrypted File System

- **فایل‌های چندجریانی**[1]: یک جریان داده، دنباله‌ای از بایت‌ها است. یک فایل چند جریانی، همانند چند فایل تک جریانی است که همگی تحت یک نام کلی در سیستم فایل ذخیره شده‌اند و هر کدام همانند یک بخش یکتا، قابل ایجاد کردن، تغییر دادن یا حذف کردن هستند. در NTFS، هر فایل می‌تواند شامل چند جریان باشد. هنگام کار کردن با فایل‌ها بر روی یک درایو غیر NTFS، تنها یک جریان از داده قابل دسترسی است و تنها محتوی فایل تلقی می‌شود. این جریان اصلی، بدون نام (Main, Unnamed Stream) و تنها جریانی است که درایوهای غیر NTFS پشتیبانی می کنند. در NTFS هر فایل علاوه بر این جریان اصلی بدون نام، می‌تواند در برگیرنده تعدادی دیگر جریان نامدار (Named Stream) نیز باشد. از این فایل‌ها بیشتر برای تبدیل فرمت‌ها[2] استفاده می‌شود.

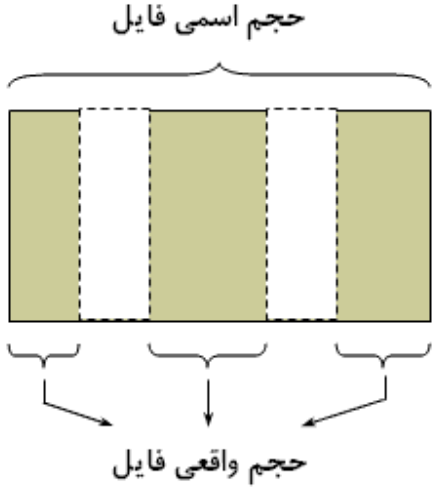


شکل ۷-۵ - فایل چندجریانی

- **فشرده‌سازی**[3]: فایل‌ها در درایوهای NTFS برای صرفه‌جویی در فضای دیسک، می‌توانند به‌صورت فشرده ذخیره شوند. فشرده‌سازی، به معنای کاستن از فضای لازم برای ذخیره‌ی اطلاعات است. پس از این‌که فایلی فشرده شد، برای خواندن اطلاعات آن، بایستی این فایل از حالت فشرده خارج شود[4].
- **سهمیه‌های دیسک**[5]: با این ویژگی می‌توانید علاوه بر کنترل فضای دیسک که هر کاربر استفاده می‌کند، بیشینه‌ی فضای در اختیار او را مشخص کنید.

[1] Multiple file stream
[2] Convert
[3] Compression
[4] Decompression
[5] Disk Quotas

- **فایل‌های تُنُک**[1]: فایل تنک، فایل معمولا بزرگی است که اطلاعات معنادار کمی در آن ذخیره شده است. NTFS سازوکاری را برای برنامه ها تدارک دیده تا بتوانند تنها پاره‌های معنادار فایل (شامل داده های غیر صفر) را واقعا بر روی دیسک ذخیره کنند و فضایی به پاره های صفر اختصاص داده نشود. این خاصیت بسیار مؤثرتر از فشرده کردن فایل است چون هیچ فضا و زمان اضافه‌ای صرف فشرده کردن یا از حالت فشرده خارج کردن این پاره‌ها نمی‌شود. مثلا فایلی خواهیم داشت با اندازه‌ی اسمی ۴ گیگابایت در حالی که واقعا چند کیلوبایت از دیسک را اشغال کرده است. بخش‌های صفر در فایل‌های تنک فضایی را بر روی دیسک اشغال نمی‌کنند.



شکل ۷-٦ - ساختار فایلهای تنک

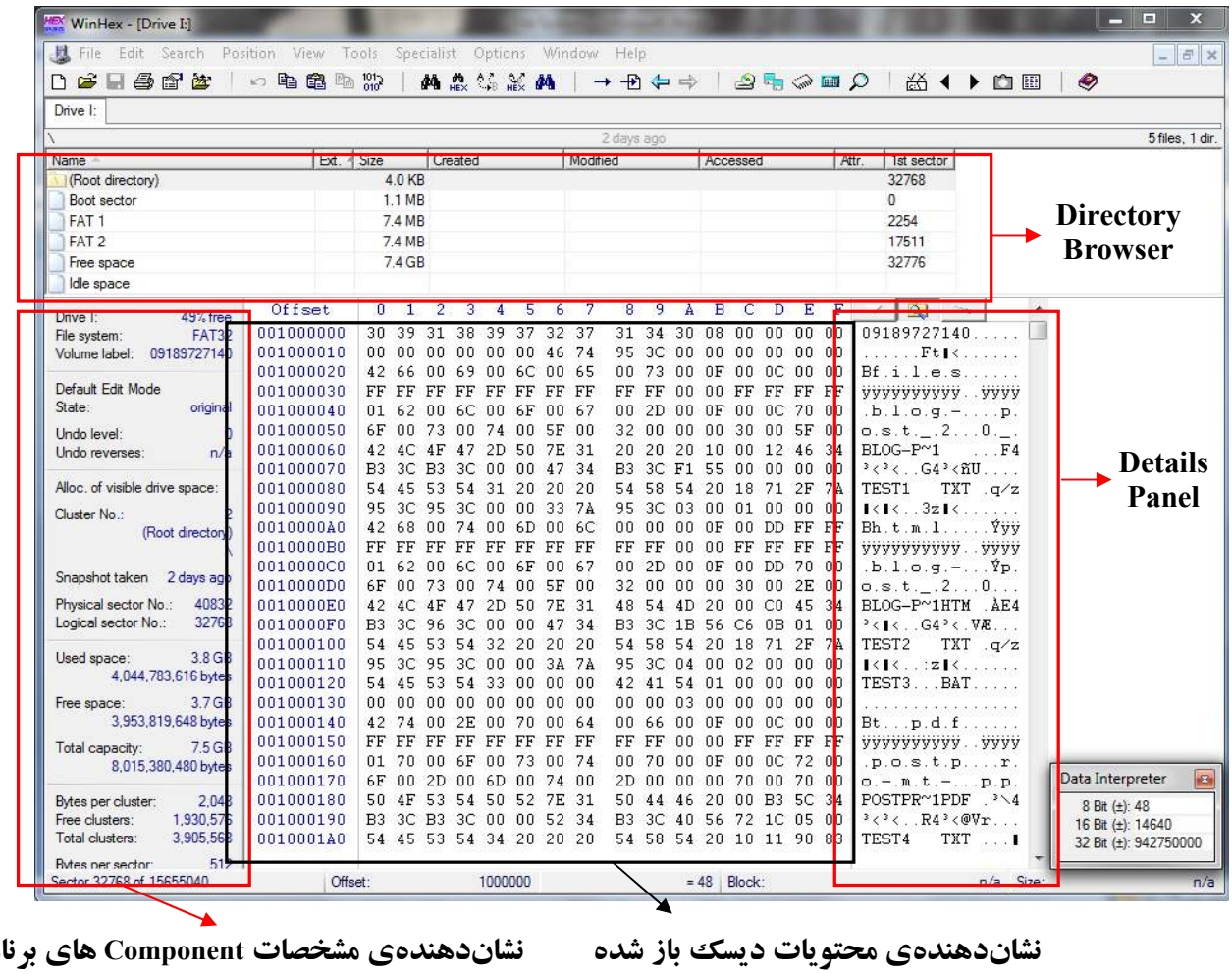
- **سرویس نمایه‌سازی**[2]: این سرویس، امکان جستجوی سریع درون فایل‌ها به‌دنبال کلمات‌کلیدی و عبارات به‌علاوه مرتب‌سازی و جستجوی فایل‌ها با توجه به نام یا سایر خصوصیات[3] نظیر زمان ایجاد شدن را سهولت و سرعت می‌بخشد. هرچند این سرویس به تنهایی از ویژگی‌های NTFS نیست ولی کارایی کامل آن تنها بر روی درایوهای NTFS به دست می‌آید.
- **چک خطای CRC**: داده‌های ذخیره شده در دیسک سخت به کمک روش تشخیص خطای CRC کنترل می‌شوند تا از بروز خطای احتمالی در آن جلوگیری شود.

همچنین ویژگی‌های دیگری نیز وجود دارد که که در این مجال امکان پرداختن به تمام آن‌ها وجود ندارد.

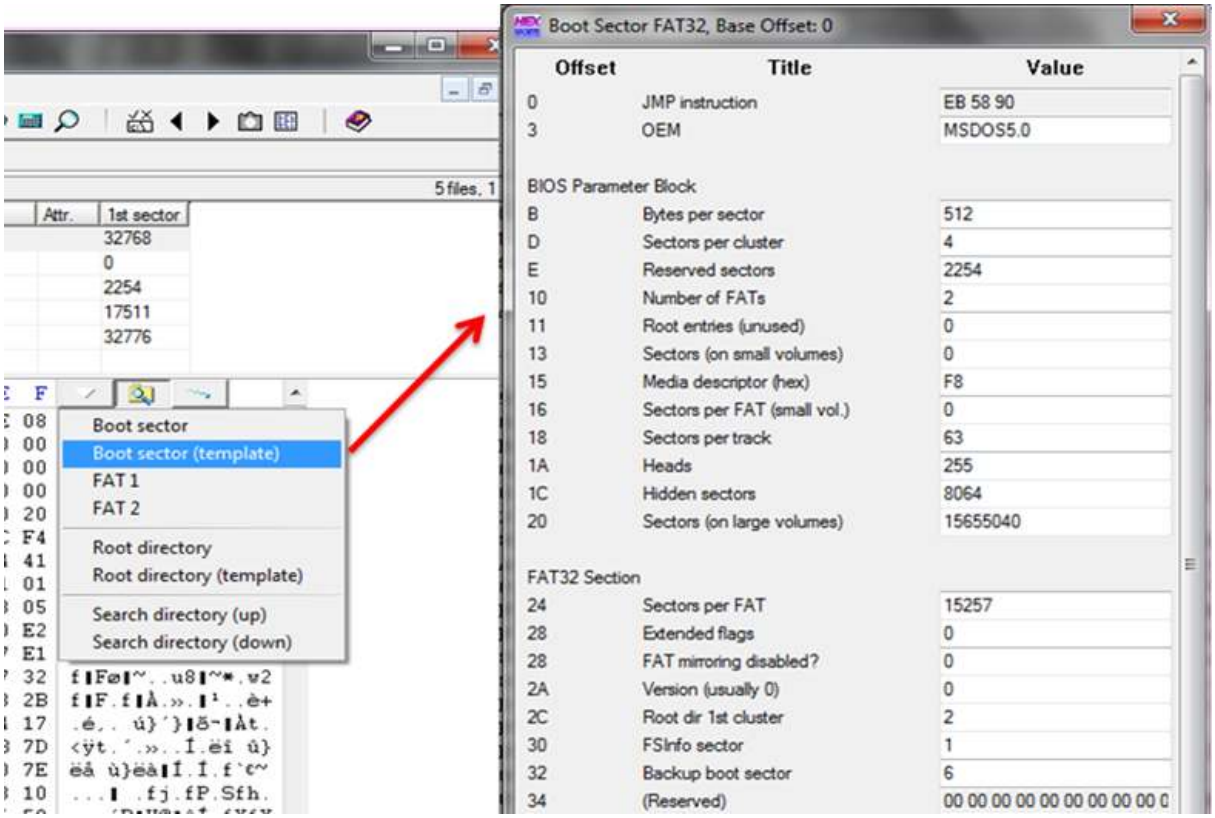
[1] Sparse files
[2] Indexing Service
[3] Attributes

۷-٤- معرفی نرم‌افزار WinHex:

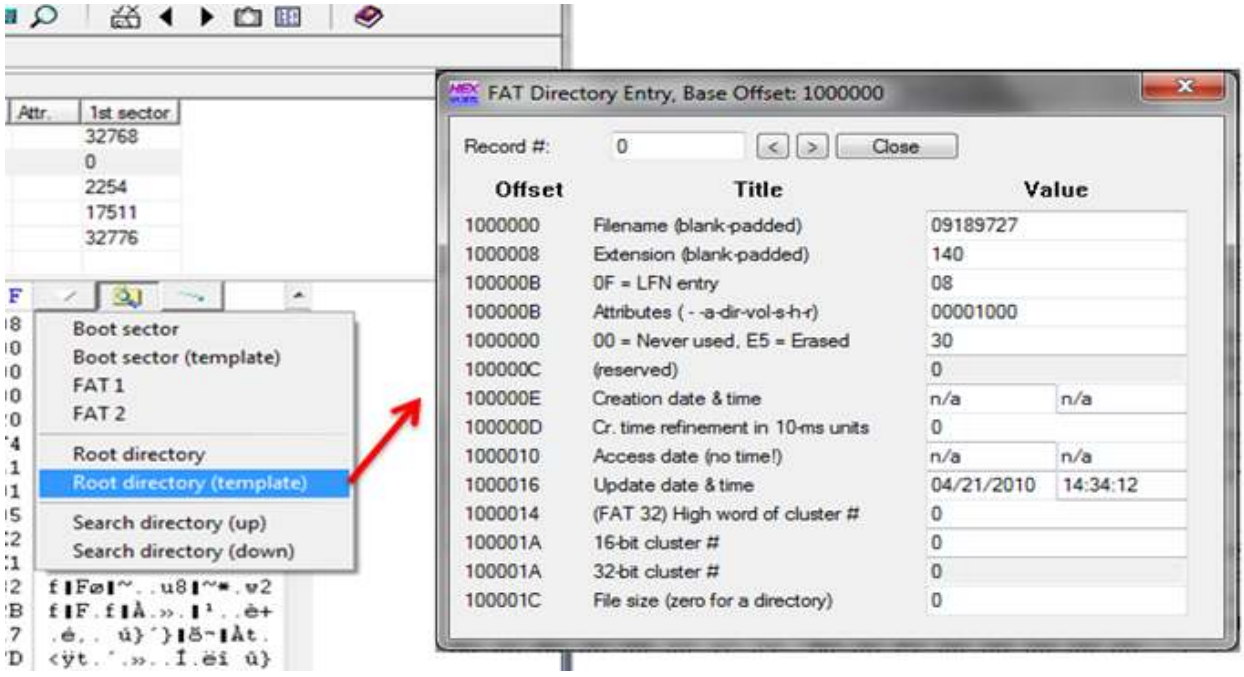
این نرم‌افزار که محصول شرکت X-Ways Software Technology AG است، نرم‌افزار مناسبی برای مشاهده و تغییر محتویات دیسک‌سخت و سیستم‌فایل‌های ویندوز است. در هنگام انجام این آزمایش از نسخه‌ی SR 15.2 استفاده می‌کنیم. در شکل‌هایی که در ادامه مشاهده می‌کنید، با برخی امکانات این نرم‌افزار که به این آزمایش مربوط است، آشنایی کلی پیدا خواهید کرد و جزییات بیشتر، در دستورکار بررسی خواهد شد.



شکل ۷-۷ - نمایی از نرم‌افزار WinHex



شکل ۷-۸- نمایی از پنجره‌ی حاوی اطلاعات Boot Sector



شکل ۷-۹ - نمایی از پنجره‌ی حاوی اطلاعات Root Directory

توجه:

با توجه به اینکه این آزمایش برروی فلش‌مموری انجام می‌شود، در هنگام آزمایش، یک فلش‌مموری به همراه داشته باشید.

۷-۵- تکلیف جلسه بعد

با استفاده از نرم‌افزار DiskPart، طریقه Bootable ساختن یک فلش‌مموری برای نصب ویندوز از طریق آن را بیان کنید.

۷-٦- مراجع

1. Silberschatz A., Galvin P.B., Gagne G., **Instructor's manual for Operating system concepts**, 8'th Ed., Wiley Publishing, Inc., 2005.
2. Tanenbaum, **Modern Operating Systems**, 2'nd Ed., Prentice Hall Publishing, Inc., 2006.
۳. گلشنی، رکسانا. امینی حاجی باشی، سحر. **تحلیل سیستم‌فایل FAT**. آزمایشگاه سیستم‌عامل گروه، کامپیوتر دانشگاه آزاد اسلامی واحد دماوند. زمستان ۱۳۸۸.
۴. میاندج، الهام. **بررسی سیستم‌فایل NTFS**. پروژه آزمایشگاه سیستم‌عامل، دانشگاه علم و صنعت بهشهر. زمستان ۱۳۸۵.

۷-۷- دستور کار

توجه ۱:

با توجه به اینکه تغییر و دستکاری فایل توسط نرم‌افزار WinHex ممکن است باعث آسیب رساندن به آن فایل‌ها شود، لذا از دستکاری فایل‌های مهم خودداری نمایید و تنها برروی فلش مموری خود آزمایش را انجام دهید. در ضمن به یاد داشته باشید که برای مشاهده تغییرات اعمال شده در فایل‌ها و پوشه‌ها، Refresh برنامه ضروری است.

توجه ۲:

در این آزمایش لازم است فلش مموری شما Format گردد پس قبل از شروع آزمایش، اطلاعات داخل آن‌را که ضروری هستند، در جایی ذخیره نمایید.

۱. در Run عبارت DiskPart را تایپ کرده کلید Enter را فشار دهید تا برنامه DiskPart باز شود.

الف) با استفاده از دستور Help، لیست دستورات برنامه را مشاهده کنید و۳ دستور آشنای آن را بنویسید. این دستورات چه کاری انجام می‌دهد.

ب) با استفاده از دستور List گزینه‌های ممکن برای آن نمایش داده می‌شود.هر یک از آنها را در گزارش‌کار خود یادداشت کنید.به نظر شما منظور از VDisk چیست؟

ج) با وارد کردن دستور List [Type] (مثلاً List Disk) که به جای [Type] می‌توانید لیست دیسک‌ها، پارتیشن‌ها، ولوم‌ها و دیسک‌های مجازی سیستم را مشاهده کنید. فلش مموری شما کدام یک از دیسک‌های سیستم است می‌باشد؟

با اجرای دستور Select [Type] و قرار دادن یکی از گزینه‌هایVDisk, Partition, Disk, Volume مربوطه، می‌توانید آن را به حالت فعال برای اعمال تغییرات (مثلاً فرمت کردن) تبدیل کنید. دیسک خود (فلش مموری‌تان) را از طریق دستور Select Volume [Your Disk Volume Number] انتخاب کنید.

د) با وارد کردن دستور FILESYSTEMS، می‌توانید لیست سیستم‌فایل‌هایی را که ویندوز حاظر پشتیبانی می‌کند، مشاهده کنید. مقدار Allocation Unit Size در این سیستم‌فایل‌ها، نشان‌دهنده‌ی چه مفهوم ذکرشده‌ایی در پیش‌آگاهی است؟

ه) دستور Help Format را وارد کنید. با استفاده از انواع دستور Format، فلش خود را با فایل سیستم FAT32 فرمت کنید، به‌گونه‌ایی که برچسب (Label) آن معادل "OS-Lab" باشد و برای آزمایش‌های

بعدی آماده شوید. سپس صبر کنید تا فلش مموری‌تان کاملاً فرمت گردد. دستوری را که وارد کرده‌اید، در گزارش کارتان یادداشت کنید.

۲. نرم افزار WinHex را باز کرده، به مسیر زیر بروید.

Tools → Open Disk

و فلش مموری‌تان را انتخاب کنید.

الف) Directory Browser شامل چه مواردی است؟

(**راهنمایی:** در صورتی که Directory Browser را مشاهده نمی‌کنید، از مسیر زیر آن‌را به برنامه اضافه کنید: Directory Browser → Show → View)

در پنل سمت چپ مشخصات دیسک‌تان دیده می‌شود. با توجه به آن، موارد زیر را در گزارش کار خود یادداشت کنید.

ب) شماره کلاستری که Root Directory در آن واقع شده است.

ج) تعداد کل سکتورهای فیزیکی و منطقی

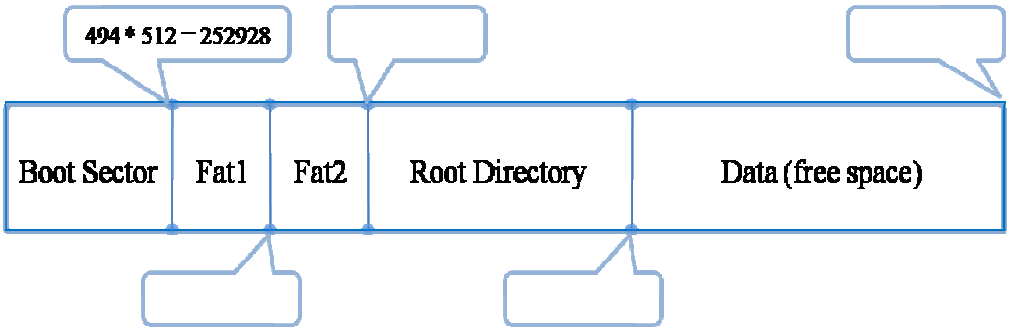
د) حجم اختصاص داده شده به هر کلاستر

ه) هر کلاسور شامل چند سکتور است؟

۳. با استفاده از ستون 1st sector (شماره اولین سکتور) و داده‌هایی که در سوال قبل به دست آورده‌اید، ساختار سیستم فایل FAT را براساس الگوی زیر در گزارش کار کامل کنید.

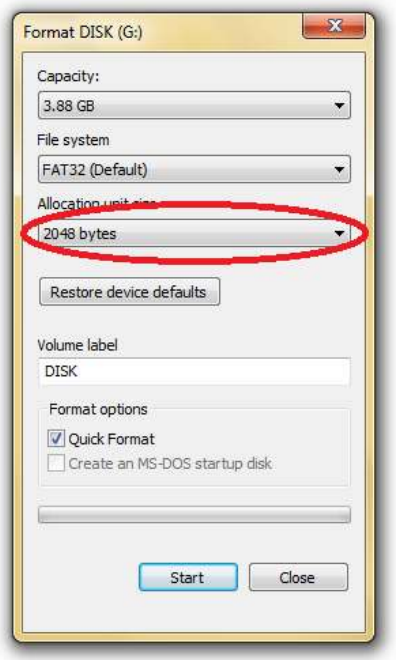
(**راهنمایی:** تعداد سکتورها را در هر یک از موارد زیر به دست آورده و در تعداد بایتها در هر سکتور ضرب کنید.)

Boot Sector Length = Byte Per Sector × Boot Sector Number



آیا مقادیری که محاسبه کرده‌اید، با مقادیری که توسط برنامه WinHex مشاهده می‌کنید، یکسان است؟ (مثلاً روی FAT1 کلیک کنید و مقدار Offset نشان‌داده شده را از مقدار Hex به Decimal تبدیل کرده و با مقداری که محاسبه نموده‌اید، مقایسه نمایید. تبدیل مقادیر Hex به Decimal را می‌توانید با استفاده از امکانات برنامه و با زدن دکمه‌ی F8 انجام دهید)

۴. با کلیک راست بر روی دیسک و انتخاب گزینه‌ی فرمت، می‌توانید طول کلاستر در هر دیسک را مشخص (یا مشاهده) کنید.



الف) طول کلاستر فلش مموری شما چقدر است؟

حال می‌خواهیم طول کلاستر را تغییر دهیم. قبل از اینکه به قسمت (ب) مراجعه کنید، با توجه به مطالب ذکرشده در پیش‌آگاهی، فکر می‌کنید این مقدار در چه قسمتی از دیسک تعیین می‌گردد.

ب) پنجره‌ی حاوی اطلاعات Boot Sector را باز کنید (چگونگی باز کردن این پنجره را در شکل ۶ مربوط به قسمت آشنایی با نرم‌افزار آموختید). همان‌طور که ملاحظه می‌کنید، با این فیلدها در پیش‌آگاهی (جدول ۴) آشنا شده بودید. طول کلاستر را به 2KB تغییر دهید. در گزارش‌کار نحوه‌ی انجام این کار را توضیح دهید.

ج) به نظر شما این کاهش اندازه‌ی کلاستر، چه تأثیری بر کارایی سیستم دارد؟

۵. پنجره‌ی حاوی اطلاعات Root Directory را باز کنید (با توجه به عکس ۷).

الف) همان‌طور که مشاهده می‌کنید، شماره Record اولیه که در این قسمت وجود دارد، برابر 0 است. به‌نظر شما رکورد 0 مربوط به چیست؟ (شکل ۳ را مدنظر داشته باشید و دقت کنید که اندازه‌ی فایل صفر است!) هر رکورد نشان‌دهنده‌ی چه اطلاعاتی است؟ (از عنوان پنجره‌ی حاظر کمک بگیرید). هر رکورد شامل چند بایت است و چرا؟ در حال حاظر چند رکورد از فلش‌مموری شما اشغال گردیده است؟ (شماره رکورد را افزایش دهید تا جایی‌که تمام بیت‌ها صفر گردد). با توجه به کدام فیلد و چه مقداری می‌توان فهمید که رکورد مربوط به یک فایل حذف شده است؟

خارج از نرم‌افزار و توسط Windows Explorer فایلی با نام Test خارج از فلش مموری ایجاد کنید و سپس در فلش‌مموری‌تان قرار دهید (به‌منظور رسیدن به اهداف آموزشی این آزمایش، توجه کنید که در داخل فلش‌مموری‌تان این فایل‌ها را ایجاد نکنید!). به برنامه WinHex بازگشته و آن‌را از مسیر زیر Refresh کنید.

View → Refresh View

ب) با توجه به مقایسه نسبت به قسمت قبل، در گزارش‌کار خود، ذکر کنید که چند رکورد به اطلاعات Root Directory افزوده شده است؟ و با توجه به Attribute آن‌ها و کمک گرفتن از شکل ۴، ذکر کنید که هر رکورد حاوی چیست؟

ج) کدام رکورد حاوی پسوند فایل Test است؟ مقدار آن‌را تغییر دهید (مثلاً تبدیل به BAT کنید) و سپس تغییرات را ذخیره کنید (Ctrl + S). آیا با رفتن به مکان این فایل در فلش‌مموری، تغییر نوع را مشاهده می‌کنید؟

د) فایل Test را توسط همین نرم‌افزار حذف کنید.

(**توجه:** به تغییر نام فایل، هنگامی‌که تنظیمات حذف را انجام می‌دهید، برای انجام قسمت بعدی آزمایش، دقت کنید!). با مشاهده‌ی محتویات فلش‌مموری خود، از حذف این فایل مطمئن شوید و مکانیزم حذف فایل را در گزارش‌کار خود ذکر کنید.

ه) آیا می‌توانید دوباره فایل را بازگردانید؟ چگونه؟

۶. مدخل دایرکتوری را آورده شماره رکورد را آن‌قدر زیاد کنید که همه مقادیر صفر شود (اولین رکورد خالی).

الف) با توجه به آن‌چه تاکنون آموخته‌اید، سعی کنید با تغییر در آن یک فایل جدید به نام Test1 از نوع Read-Only و با پسوند txt ایجاد کرده و ذخیره کنید. به دیسک خود بروید چه مشاهده می‌کنید؟

ب) آیا فایل ایجاد شده دارای پسوند txt است؟ اگر نیست، حدس می‌زنید دلیل چه باشد؟ سعی کنید آن را برطرف کنید و به مربی نشان دهید. اگر نتوانستید، به‌عنوان تکلیف مکانیزم را برای جلسه آینده به همراه بیاورید.

۷. در قسمت Directory Browser، برروی FAT1 کلیک کرده و محتویات آن‌را بیاورید.

الف) چند بایت اول آن رزرو است؟ شماره‌ی کلاستر(ها)یی که Root Directory در آن واقع است، چند است؟

ب) چند بایت در این جدول، برای هر کلاستر تخصیص داده شده است؟ چرا؟

ج) حال مشابه دستورکار ۵ – قسمت (ب) این آزمایش، یک پوشه به نام Test ایجاد کرده و داخل فلش‌مموری قرار دهید.

پس از Refresh کردن نرم‌افزار به سئوالات زیر پاسخ دهید:

a) چند رکورد از Root Directory را تحت تأثیر قرار داده است؟

b) غیر از مقادیر تاریخ و مدت زمان ایجاد، در چه پارامترهایی با فایل Test متفاوت است؟ تفاوت‌ها را تفسیر کنید.

c) چه کلاستری به این دایرکتوری جدید اختصاص داده شده است؟

آیا همان کلاستر(ها)یی است که به Root Directory تخصیص داده شده است؟

۸. فایلی را به دلخواه با حجمی بیش از 5 KB در پوشه‌ی Test قرار دهید و پس از Refresh کردن برنامه، به سئوالات زیر پاسخ دهید:

الف) با توجه به حجم کنونی کلاسترها، برای ذخیره‌ی این فایل چند کلاستر لازم است؟

ب) وارد FAT شده و برروی اطلاعات کلاستر تخصیص داده شده به پوشه‌ی Test رفته و دنباله‌ی کلاسترهایی را که اکنون به آن تعلق دارد را در گزارش‌کار خود بنویسید.

آزمایش هفتم

آشنایی با لینوکس

۸-۱- مقدمه

لینوکس به عنوان یک سیستم‌عامل Open Source تلفیقی از ایده‌های سرراستی است که در طول زمان و به تدریج به آن افزوده شده است. اجازه کار با اجزای درونی لینوکس و شناخت آن‌ها برای کاربران، سبب تسهیل درک مفاهیم سیستم‌عامل به‌خصوص برای دانشجویان درس آزمایشگاه سیستم‌عامل خواهد شد. این امکان در یک سیستم‌عامل تجاری مانند ویندوز، بسیار ضعیف‌تر تعبیه شده است.

لینوکس نیز همانند سایر سیستم‌عامل‌های پیشرفته‌ی امروزی، دارای رابط گرافیکی کاربرپسند است. اما به خصوص برای کارهای راهبری و سیستمی در لینوکس و به خصوص برای مقاصد آموزشی و برنامه‌نویسی آن، لازم است کاربر با پوسته‌ی قدرتمند و جامع کارآمد آن آشنا شود. به خصوص در محیط تجاری و کار نیز که بسیاری از عملیات را راهبران تحت خط فرمان انجام می‌دهند، چنین نیازی ملموس‌تر درک می‌شود. بنابراین در این آزمایش‌ها تلاش عمده بر ارائه مفاهیم به صورت پایه‌ای و خط فرمان بوده و اتکای کمتری بر رابط گرافیکی که تنوع و تفاوت زیادی در توزیع‌های مختلف لینوکس دارد، بوده است.

۸-۲- هدف

هدف در این فصل، آشنا ساختن مقدماتی دانشجو با لینوکس و نحوه کار با آن در حدی است که زمینه برای انجام برخی آزمایش‌ها تحت آن فراهم شود. در این فصل موارد زیر را خواهید آموخت:

- شناخت از مزایای لینوکس
- شناخت از رایج‌ترین نسخه‌های لینوکس با درک تفاوت آن‌ها
- دانش مبنایی از سیستم‌پرونده لینوکس و تفاوت آن با ویندوز
- دانش مبنایی از نحوه ایجاد و کار با پرونده‌ها در لینوکس و عملیات اصلی بر آن‌ها و معرفی رایج‌ترین ویرایشگر لینوکس
- آشنایی با دستورات پایه مهم کار با لینوکس

۸-۳- پیش‌آگاهی

۸-۳-۱- تاریخچه

شروع به کار اصلی لینوکس در حدود سال ۱۹۹۱ بود. زمانی‌که دانشجوی فنلاندی به نام Trovalds از کار کردن با سیستم عامل Minix احساس راحتی نمی‌کرد. Minix نسخه آموزشی Unix بود که Unix در آن زمان برای کار با Serverها مورد استفاده قرار می‌گرفت و هزینه خرید بالایی داشت. تصمیم برای ایجاد یک Kernel بهتر، سبب ایجاد سیستم‌عامل Linux شد. شاید هوشمندانه‌ترین عمل Trovalds آن بود که تصمیم نداشت همه چیز را در این زمینه به تنهایی انجام دهد. وی افرادی را که مایل به همکاری با او در زمینه ایجاد ویرایش بهتری از Minix بودند از طریق پیامی در Usenet (شبکه ارتباطی اصلی در اینترنت قدیمی مبتنی بر خط فرمان برای تبادل اطلاعات و کمک به سایرین مبتنی بود) یافت و نسخه‌های رایگان متعددی مبنی بر این تلاش، ایجاد و پیشنهاد شد.حمایت نهادهایی مانند FSF یا Free Software Foundation از این سیستم‌عامل جدید و نرم‌افزارهایی که برای استفاده رایگان تحت آن عرضه شده بودند(GNU Software)، به رواج آن کمک زیادی کرد. Trovalds بعدها برای لینوکس، امتیاز GPL یا General Public License را نیز گرفت تا هر کس بتواند آن را در اختیار داشته باشد و دستکاری کند و بهبود دهد. در ادامه به سه توزیع[1] اصلی سیستم‌عامل لینوکس که رایج‌ترین هستند اشاره‌ای خواهد شد

[1] Distribution

(RedHat, SUSE, Ubuntu) و لازم به ذکر است که آزمایشات تدوین شده برای هر سه توزیع این سیستم‌عامل قابل استفاده هستند، که تقریبا ۹۰٪ از همه نسخ مورد استفاده از لینوکس را در سراسر جهان شامل می‌شوند.

۸-۳-۲- توزیع‌های لینوکس

۸-۳-۲-۱- Red Hat

نسخه‌ای که در کارولینای شمالی تهیه شده است و نقش عمده‌ای در راه یافتن لینوکس به عرصه‌ی Data Centerهای بسیاری از شرکت‌ها داشته است. علت اصلی این موفقیت، خدمات پشتیبانی‌ای بود که از طرف شرکت Red Hat برای این نسخه از لینوکس فراهم شد. در وهله اول، چنین خدماتی به حمایت از نرم‌افزارها و سخت‌افزارهای مختلف برمی‌گشت. بدین معنی که کاربران در کار با نرم‌افزارها و سخت‌افزارهای متنوع از بابت استفاده از لینوکس، آسوده‌خاطر بودند و در وهله دوم شرکت Red Hat با افزودن خدمات تجاری، حاضر به کمک به کاربران نیز بود.

امروزه سه نسخه که برمبنای Red Hat توسعه یافته‌اند، عرضه می‌شوند. اولی که مهمترین آن‌هاست RHEL یا Red Hat Enterprise Linux است که در دو نسخه Server و یک نسخه Desktop عرضه شده است. این نسخه تجاری است و بنابراین امکان دانلود رایگان ندارد، اما هنوز Open Source است. علت پولی شدن این نسخه،تنها افزوده شدن لوگوی شرکت RedHat است.

دومین توسعه‌ی Red Hat، به پروژه‌ی Fedora Open Source باز می‌گردد. Fedora به عنوان محیط تولید و توسعه RHEL به حساب می‌آید و بسیاری از قطعات نرم‌افزاری، ابتدا برای Fedora تست و استفاده می‌شوند و اگر آزمایش موفقیت‌آمیزی داشتند، به REHL انتقال داده می‌شوند. نسخه Fedora، از آدرس www.redhat.com/fedoraبه صورت رایگان قابل دانلود است.

از آنجا که نسخه RHEL فقط به خاطر استفاده از یک لوگو، پولی شده وگرنه کاملا متن‌باز و رایگان است، CentOS (Community ENTerprise Operating System) با برداشتن چنین لوگویی، نسخه‌ی رایگان RHEL را به صورت رایگان عرضه کرده است. برای دریافت چنین نسخه‌ای می‌توان به وب‌سایت www.centos.org مراجعه کرد.

۸-۳-۲-۲- SUSE

این توزیع در آلمان پا به عرصه وجود نهاد و به دلیل عرضه بسته‌های نرم‌افزاری مختلف از ابتدا به همراه این توزیع، به سرعت فراگیر شد. SUSE از اولین نسخی بود که با انگیزه مادی مطرح شد و تنها پیش‌ارائه‌ی[1] آن به صورت رایگان قابل عرضه بود. شرکت Novell که خود یک سیستم‌عامل آشنا عرضه نموده است که اکنون منسوخ شده، در سال ۱۹۹۴، امتیاز SUSE را خرید و آن را به توزیعی در حد سازمان‌ها[2] یعنی قابل رقابت با RedHat رساند.

امروزه دو جهت‌گیری متفاوت از SUSE وجود دارد، SUSE Linux Enterprise که یک سیستم‌عامل تجاری به حساب می‌آید وپشتیبانی به همراه دارد و خود در نسخ مختلف ارائه می‌شود که برای کاربردهای متنوع مانند بلادرنگ[3] نیز قابل تنظیم است و نسخه SUSE Linux Enterprise Desktop که البته این نسخه را می‌توان به صورت رایگان از آدرس www.novell.com/downloads تهیه نمود.

جهت‌گیری دیگر SUSE، OpenSUSE نام دارد که یک محصول کاملا متن باز[4] است. این ویرایش نه‌تنها ویرایشی پایا[5] به حساب می‌آید، بلکه محیط توسعه‌ی مناسبی برای ایجاد نرم‌افزارهای جدید نیز به حساب می‌آید. می‌توان این توزیع را از آدرس www.opensuse.com دانلود نمود.

۸-۳-۲-۳- Ubuntu

این توزیع، نسبتا جدید به حساب می‌آید و سرآغاز بسیار موفق آن از آفریقای جنوبی و یک فرد میلیونر آن‌جا رقم خورد. این توزیع بسیار کاربرپسند بود و CD آن برای استفاده‌های شخصی[6] به صورت رایگان از وب‌سایت www.ubuntu.com برای همه قابل دانلود است. اضافه بر نسخه Desktop رایگان، نسخه Server از Ubuntu نیز به صورت رایگان قابل استفاده است و به سرعت در حال رواج می‌باشد. البته کاربرانی که تمایل به پشتیبانی دارند، می‌توانند آن را از شرکت Canonical که شرکتی جدا برای ارائه خدمات حرفه‌ای پشتیبانی Ubuntu است، خریداری کنند. نکته قابل توجه در رابطه با Ubuntu آن است که هر شش ماه یک نسخه جدید از آن عرضه می‌شود.

---

[1] Demo
[2] Enterprise
[3] Real-Time
[4] Open Source
[5] Stable
[6] Desktop

۸-۳-۳- آشنایی با لینوکس

۸-۳-۳-۱- خصوصیات لینوکس

خصوصیات متعددی در رابطه با لینوکس که امروزه به همراه ویندوز مایکروسافت، نقش برجسته‌ای در استفاده‌ی کاربران از سیستم‌عامل دارد، قابل بیان است که در زیر به اجمال به برخی از کلیدی‌ترین آن‌ها پرداخته می‌شود. اما از متمایزترین و مهم‌ترین مزایای آن، رایگان بودن و متن باز بودن آن است.

- **طراحی مستقل از سخت‌افزار:** از آن‌جا که قسمت اعظم این سیستم‌عامل به جای آن‌که با اسمبلی نوشته شده باشد، با زبان C نوشته شده است، دارای انعطاف و اطمینان زیادی است و امروزه بر بسترهای سخت‌افزاری مختلف قابل نصب و استفاده است. همچنین نگه‌داری آن نیز ساده است و به همین دلیل، تاکنون توزیع‌های مختلفی از آن عرضه شده است. البته این را شاید بتوان یک ایراد نیز به حساب آورد، زیرا نسبت به سیستم‌عامل‌هایی که با کد اسمبلی پیاده‌سازی شده‌اند، از لحاظ اجرا کندتر است. بر خلاف نسخه‌های متعدد ویندوز که در شناسایی سخت‌افزارهای مختلف نیاز به نصب راه‌انداز[1] مربوطه دارند، لینوکس بخش عمده‌ای از سخت‌افزارها را بدون نصب راه‌انداز، شناسایی و قابل استفاده می‌کند.
- **چندکاربره بودن[2]:** در یک زمان، یک یا چند کاربر می‌توانند از کامپیوتری که دارای سیستم‌عامل لینوکس است استفاده کنند و منابع سیستم را با لحاظ نمودن تمهیداتی جهت محدود کردن کاربران از آسیب‌رساندن به همدیگر، به اشتراک بگذارند. این خصوصیات از ویژگی‌های لازم هر سیستم‌عامل پیشرفته کنونی به حساب می‌آید.
- **چندوظیفه‌ای[3] بودن:** همانند سایر سیستم‌عامل‌های پیشرفته‌ی کنونی، لینوکس نیز اجازه‌ی اجرای همزمان چندین برنامه را به کاربر می‌دهد.

۸-۳-٤- سیستم‌پرونده در لینوکس

هر چیزی که در سیستم‌عامل لینوکس مورد استفاده قرار می‌گیرد یا ذخیره می‌شود به نوعی تحت عنوان یک پرونده بر دیسک سخت ذخیره می‌شود. برنامه‌ها و پرونده‌ها و تمامی منابع (جز ارتباطات شبکه) در

[1] Driver
[2] Multiuser
[3] Multitasking

لینوکس به عنوان پرونده شناخته می‌شوند. بنابراین سیستم پرونده در لینوکس، مفهومی متمایز با سیستم پرونده در ویندوز دارد. پرونده سیستم در لینوکس در واقع تقریبا تمام منابع را در برمی‌گیرد.
بنابراین انتقال یک داده به یک دستگاه جانبی مانند چاپگر، در لینوکس به نوعی نوشتن در یک پرونده به حساب می‌آید. یا اینکه به عنوان مثال دیسک سخت را لینوکس مانند یک پرونده با آدرس /dev/hda یا /dev/hdb می‌شناسد.

نقش فهرست‌ها[1] نیز مشابه است. در واقع فهرست‌ها خود پرونده‌هایی به حساب می‌آیند که در آن‌ها اطلاعات مربوط به سازماندهی پرونده‌های دیگر ذخیره می‌شود. فهرست‌ها یا Directoryها در واقع همان نقشی را در لینوکس ایفا می‌کنند که پوشه‌ها یا Folderها در ویندوز برعهده دارند. بدین ترتیب یکی دیگر از نقطه تمایزهای سیستم‌پرونده در لینوکس نسبت به ویندوز آن است که همه‌ی پرونده‌ها و منابع در لینوکس، در داخل فهرست‌ها قرار می‌گیرند. قوانین عمومی زیر در رابطه با تمام توزیع‌های لینوکس برقرار است:

- فهرست‌ها می‌توانند متداخل یا تودرتو باشند و در عمل محدودیتی برای این تودرتویی وجود ندارد
- فهرست‌ها با علامت اسلش یا / از هم جدا می‌شوند. این درحالی است که ویندوز از Backslash یا \ برای تفکیک فهرست‌ها استفاده می‌کند.
- ریشه یا Root در هرفهرست با علامت اسلش اول نمایش داده می‌شود. به عنوان مثال اگر بخواهید محتویان فهرستی با نام etc را که پرونده‌های پیکربندی سیستم در آن قرار دارند ببینید، دستور ls /etc را در خط فرمان خواهید نوشت. استفاده از کاراکتر اسلش به سیستم‌عامل می‌گوید که در فهرست ریشه به دنبال فهرست etc بگردد تا محتوی آن را نمایش دهد.

۸-۳-٤-۱- فهرست‌های رایج

در شروع کار با لینوکس، دانستن مکان و محتوی فهرست‌های رایج، مفید است. لیست زیر حاوی مکان و استفاده‌ی برخی از رایج‌ترین فهرست‌های لینوکس است. تمام این فهرست‌ها به صورت پیش‌فرض و همزمان با نصب لینوکس، ایجاد می‌شوند:

- /boot هسته لینوکس و پرونده‌های پشتیبان آن در این فهرست نگهداری می‌شوند.

[1] Directories

- dev/ پرونده‌های دستگاه‌ها را در خود دارد. دستگاه‌های فیزیکی در لینوکس توسط پرونده‌ها نمایش داده می‌شوند. بنابراین دیسک سخت شما، چاپگر یا حافظه متصل به USB، دارای پرونده‌های معادل در این فهرست هستند.
- etc/ مسیر پرونده‌های پیکربندی سیستم است که این پرونده‌ها، اجازه‌ی استفاده‌ی خودکار کامپیوتر از شبکه، چاپگر و سایر دستگاه‌ها را زمانی که آن‌ها را پیکربندی کرده باشید، فراهم می‌کند.
- home/ فهرست حساب‌های کاربری کاربران. به عنوان مثال اگر کاربری به نام ali در سیستم ایجاد کرده باشید، آن نام کاربری دارای فهرستی با مشخصه‌ی home/ali/ خواهد بود و مکان پیش‌فرض این کاربر، زمانی که در سیستم Login کرده باشد، همین است.
- proc/ یک شبه فهرست است که به صورت فیزیکی بر دیسک واقع نشده است، اما نمایی به هسته‌ی لینوکس را فراهم می‌کند. یعنی لینوکس از طریق این فهرست، به کاربر اجازه می‌دهد تا برخی ساختمان‌داده‌های داخلی آن را مشاهده کند و در بعضی موارد نیز، آن‌ها را از طریق پرونده‌های موجود در این فهرست، کنترل کند. البته معمولا کاربران از چنین فهرستی استفاده نمی‌کنند و تنها برای حالات پیشرفته و حرفه‌ای به آن ارجاع داده می‌شود. این فهرست توسط کاربران معمول قابل دسترسی نیست مگر اینکه از امتیاز کاربر ریشه[1] یا ابرکاربر[2] استفاده کنند.
- tmp/ پرونده‌های موقت که آن‌ها را می‌توان پس از استفاده حذف کرد، در اینجا قرار می‌گیرند. به عنوان مثال، پرونده‌ها متعددی را لینوکس در زمان اجرا و استفاده از برنامه‌های سودمند، ایجاد می‌کند تا آن‌ها را ساده‌تر مدیریت کند. چنین پرونده‌هایی در این فهرست ریخته می‌شوند.
- usr/ فهرست‌های اضافه که برنامه‌های سیستمی و کاربردی، پرونده‌های کتابخانه‌ای[3] و پرونده‌های پشتیبانی را ذخیره می‌کنند. در ادامه جزییات بیشتری از آن بیان شده است:
  - user/bin/ برنامه‌های کاربر و برنامه‌های سودمند. به عنوان مثال، برنامه‌ای مانند مرورگر وب Firefox یا دستوری مانند ls که برای فهرست گرفتن از محتویات یک

[1] User Root
[2] Super User
[3] منظور از library files یا پرونده‌های کتابخانه‌ای، پرونده‌هایی کوچکی هستند که توابعی فقط خواندنی را در خود دارند که ممکن است به صورت همزمان توسط چند برنامه مورد استفاده قرار گیرند. چنین پرونده‌هایی در ویندوز DLL نام دارند.

پوشه است را می‌توان نام برد. بسیاری دیگر از برنامه‌های کاربردی و سودمند نیز به صورت پیش‌فرض در همین فهرست ذخیره شده‌اند.

- /user/sbin برنامه‌های کاربردی سیستمی و برنامه‌های سودمند. این فهرست شبیه به /user/bin است، اما برنامه‌های کاربردی و سیستمی را که برای راهبری سیستمی نیاز است، در خود دارد. این فهرست توسط کاربران معمول قابل دسترسی نیست مگر اینکه از امتیاز کاربر ریشه استفاده کنند.
- /user/local برنامه‌های جانبی و برنامه‌های دیگر کاربر، پرونده‌های پیکربندی و پایگاه داده و غیره در این فهرست ذخیره می‌شوند. این فهرست معمولا برای برنامه‌هایی است که به صورت پیش‌فرض، همراه با سیستم‌عامل نصب نمی‌شوند.

- /var پرونده‌های که با زمان متغییر هستند. این فهرست محتوی پرونده‌هایی است که سیستم عامل ایجاد یا حذف می‌کند یا می‌خواند یا می‌نویسد. به عنوان مثال در زمان چاپ یک سند، پروندهی موقت که متن را همراه با قالبش نگه می‌دارد، در فهرست /var/spool می‌ریزد. این پرونده پس از انجام عمل چاپ، پاک می‌شود.

۸-۳-٤-۲- دستورات کار با سیستم‌پرونده

همانطور که در بخش پیش اشاره شد، امکانات نسبتا متفاوت اما متنوعی در رابطه با سیستم‌پرونده در لینوکس برای کاربران فراهم شده است. برخی از مهم‌ترین دستورات پایه‌ای که برای ایجاد، حذف، پیمایش و ... سیستم‌پرونده لینوکس در اختیار قرار داده شده‌اند در جدول زیر آمده‌اند. همچنین برای آشنایی بیشتر، یک مثال از نحوه استفاده یکی از دستورات نیز ذکر شده است:

نام دستور	کاربرد
تغییر فهرست جاری	**cd** path
بازگشت به فهرست پدر	**cd ..**
تغییر مسیر به مسیرکاربر ذکر شده پس از علامت ~ به شرط داشتن اجازه دسترسی	**cd ~user**
گرفتن لیست پرونده‌ها وفهرست‌ها	**ls** path
انتقال پرونده‌ها و فهرست‌ها	**mv** path/filename newpath
کپی کردن پرونده‌ها و فهرست‌ها	**cp** source-file target-file

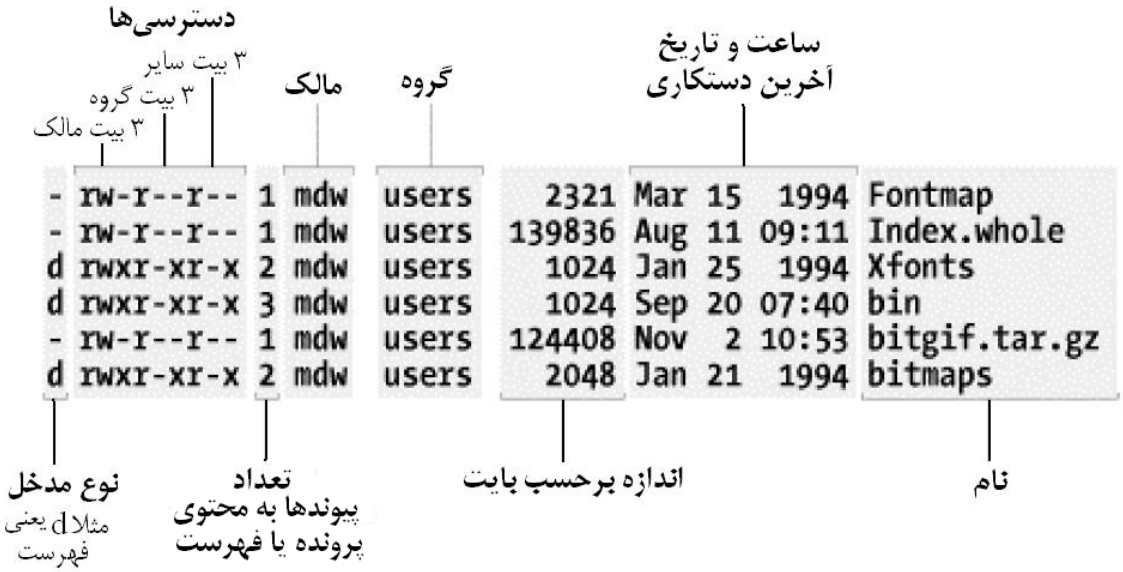
rmdir directory-name	حذف شاخه‌ها
mkdir directory-name	ایجاد شاخه‌ها
find path **–name** filename **–print**	یافتن پرونده‌ها و نمایش آن‌ها
cat path/filename	دیدن محتویات پرونده‌ها
rm path/filename	حذف پرونده‌ها
more path/filename	دیدن صفحه به صفحه‌ی پرونده‌ها
Pwd	یافتن مسیر کنونی که در آن هستیم
who	کاربرانی را که Log in کرده‌اند، همراه با پایانه‌ی مربوط به ورودشان نشان می‌دهد

مثال:

ls /home

مثال:

حاصل دستور ls –l که سوییچ l– سبب نمایش جزییات پرونده‌ها و فهرست‌های دستور می‌شود، خروجی‌ای همانند زیر دارد که تفسیر هر بخش آن در ادامه آمده است:



اجازه‌های[1] کار با فایل‌ها در سطوح اجرا (eXecute)، خواندن (Read)، نوشتن (Write) و غیرمجاز (-) تعیین شده است. نوع مدخل اگر d باشد به فهرست اشاره می‌کند و اگر – باشد نشانگر یک فایل استاندارد است و اگر l باشد بیانگر یک میانبر[2] به فایل دیگر یا در اصطلاح لینوکس یک لینک سمبولیک است.
برای تغییر اجازه‌های کار با فایل‌ها می‌توان از دستور chmod استفاده کرد. لازم به ذکر است که این دستور را تنها در حالت root مجاز به استفاده هستیم. برای این منظور باید اجازه‌ها را برای سه موجودیت، مالک (Owner or User) فایل، گروه کاربران مرتبط با فایل (Group) و سایرین (Other) تنظیم کرد. دستور chmod را می‌توان در دو حالت نسبی و مطلق استفاده نمود. در حالت مطلق، سه رقم برای تعیین نوع اجازه‌ی کار با فایل، استفاده می‌شود. برای خواندن یا Read عدد ۴، برای نوشتن یا Write عدد ۲ و برای اجرا یا eXecute، عدد ۱ منظور شده است. جمع چنین اجازه‌هایی تعیین کننده‌ی عددی است که با دستور chmod برای تعیین سطح دسترسی به فایل، استفاده می‌شود.

مثال:

Chmod 755 /somefile

دستور فوق به کاربر (مالک) همه‌ی امکان‌های خواندن، نوشتن و اجرا را می‌دهد، در حالی که برای گروه و سایرین، تنها امتیازهای خواندن و اجرا را مجاز می‌دارد.

با استفاده از حالت مطلق اجازه‌ها برای دستور chmod، عملا تمام اجازه‌های پیشین منسوخ و جایگزین می‌شود، اما اگر بخواهیم تنها برخی از اجازه‌های قبلی را تغییر دهیم، می‌توان از chmod در حالت نسبی استفاده کرد. در اینجا سه حرف، نقش مورد نظر را ایفا خواهند کرد؛ ابتدا باید تعیین شود چه کسی اجازه‌هایش تغییر خواهد کرد: u بای مالک یا کاربر، g برای گروه و o برای سایرین. سپس علامت مثبت + یا – برای تعیین افزودن یا لغو اجازه مورد نظر و در نهایت w،r و x برای تعیین نوع امتیاز. البته زمانی که امتیاز را بخواهیم برای همه منظور کنیم، می‌توان از ذکر g،u و o صرفنظر کرد و الخ.

مثال:

Chmod +x somefile

یعنی امتیاز اجرا برای همه مجاز شد.

[1] Permission
[2] Shortcut

۸-۳-٤-۳- دستورات و نکات ضروری

در هنگام کار با لینوکس در عمل در حال کار با کنسول هستید، که ممکن است گرافیکی یا متنی باشد. تمام توزیع‌های لینوکس، امکان کار با بیش از یک کنسول را به صورت همزمان عرضه می‌کنند که به آن نام کنسول مجازی[1] می‌دهند. این امر امکان ورود و کار همزمان چندین کاربر با یک رایانه شخصی را که سیستم‌عامل لینوکس بر آن نصب شده و فعال است می‌دهد. البته چنین امکانی به خصوص برای زمانی که کاربران از رابط غیرگرافیکی استفاده کنند مناسب است. یکی از مثال‌های رایج استفاده از چنین کنسول‌هایی، نوشتن و خطایابی برنامه‌ای در یک کنسول و اجرا و تست آن در کنسول دیگر است.

اکثر توزیع‌های لینوکس اغلب ۶ کنسول مجازی در اختیار قرار می‌دهند. اسامی آن‌ها از tty1 تا tty6 است و می‌توان آن‌ها را با کلیدهای Ctrl+Alt+ Function Key فعال کرد. به عنوان مثال برای رفتن به کنسول مجازی چهارم از Ctrl+Alt+ F4 استفاده می‌شود. در محیط غیر گرافیکی، Ctrl هم لازم نیست. برای برگشتن به حالت گرافیکی از حالت متنی Ctrl+Alt+ F7 استفاده کنید.

تغییر کاربر:

به صورت عام دوسطح از دسترسی برای کاربران جهت ورود به سیستم وجود دارد: کاربر root و سایرین. بهتر است جز در موارد لزوم، از کاربر root استفاده نشود، زیرا با اختیاراتی که کاربر root دارد، ممکن است رویدادهای ناگهانی اشتباهی سبب بروز حوادث بدون برگشت شود. برای اعمال یک کاربر معمول غیر root، مانند نوشتن یک پرونده متنی یا تغییر آدرس IP، نیاز به امتیاز root نیست.

برای تغییر کاربر، در اکثر لینوکس‌ها از دستور su که مخفف substitute user است استفاده می‌شود. نحوه استفاده نیز ساده است:

```
su user
```

که کاربر جاری را به کاربری با نام user تغییر می‌دهد. اگر کاربر جاری غیر از ریشه باشد، نیاز است که رمز عبور وارد کند. اگر su را به تنهایی و بدون مشخص کردن کاربر مقصد وارد کنیم، نشان دهنده تمایل به سوییچ به حالت root است.

نکته ۱: در Ubuntu، نمی‌توان بار اول su را به این صورت استفاده کرد. بلکه از sudo استفاده می‌کنند که امکان اجرای دستورات را با اجازه‌ی root فراهم می‌کند. ترفند زیر سبب خواهد شد که بتوان با su در Ubuntu، کاربران را تغییر داد:

```
sudo su
```

[1] Virtual Console

نکته۲: برای اجرای دستورات با اجازه root، باید از دستور sudo به عنوان پیشوند استفاده کرد. مثلا دستور ls / –l که تمام فایل‌ها و فهرست‌های فهرست جاری و زیرفهرست‌ها را نشان می‌دهد، برای کاربران معمولی با پیام خطای دسترسی در برخی موارد مواجه خواهد شد را باید به شکل sudo ls / -l نوشت.

ایجاد کاربر:

باید تحت امتیاز root، دستور زیر را وارد کنید:

adduser

دستور cat: رویت محتویات یک پرونده. استفاده به صورت cat filename.exension است، که یک مثال آن می‌تواند مورد زیر باشد:

$ cat /etc/hosts

تغییر رمزعبور:

از دستور Passwd استفاده می‌شود که اگر از دستور man برای تعیین گزینه‌های آن استفاده شود، متوجه می‌شویم که امکاناتی از قبیل برداشتن رمز عبور برای یک کاربر خاص، قفل کردن یک حساب کاربری، باز کردن قفل یک حساب کاربری و اجبار به تغییر رمز عبور برای یک کاربر در ورود بعدی، از طریق این دستور فراهم است.

نکته مهم:

با دوباره زدن کلید tab در هنگامی که بخشی از دستور نوشته شده است، سیستم‌عامل تمام گزینه‌های ممکن را برای تکمیل آن دستور پیشنهاد می‌دهد.

۸-۳-۵- راهنمای لینوکس

لینوکس راه‌های مختلفی برای کمک گرفتن پیشنهاد می‌کند، که به صورت خلاصه عبارتند از:

- استفاده از دستور man، سبب بهره‌گرفتن از مستندات مربوط به هر دستور خط فرمان در لینوکس می‌شود
- تقریباً تمام دستورات گزینه‌ی help– را می‌پذیرند، که استفاده از آن سبب نمایش خلاصه از گزینه‌های موجود قابل استفاده با دستور مورد نظر می‌شود

- پوسته‌ی Bash[1] نیز همانند همه‌ی پوسته‌های دیگر، دارای یک سری دستورات داخلی تعبیه‌شده[2] است که به محض بارگذاری پوسته‌ی Bash، در دسترس خواهند بود. برای دانستن بیشتر در رابطه با دستورات چنین پوسته‌ای می‌توان از دستور help کمک گرفت

۸-۳-۵-۱- نحوه استفاده از man

مهم‌ترین منبع اطلاعات در رابطه با دستورات لینوکس، man است که مخفف manual است، که راهنمای مختصر برنامه‌نویسان است. نحوه استفاده از آن بدین نحو است که ابتدا دستور man و سپس نام دستوری را که در رابطه با آن اطلاعات می‌خواهیم می‌نویسیم. همانند دستورات سیستم‌عامل DOS می‌توان از سوییچ‌های متعدد همراه با دستور نیز استفاده کرد.

به عنوان مثال در تشریح دستور passwd یعنی با نوشتن دستور man passwd خروجی زیر (که خلاصه نیز شده است) را خواهیم داشت:

```
PASSWD(1)                  User Commands                  PASSWD(1)

NAME
       passwd - change user password

SYNOPSIS
       passwd [options] [LOGIN]

DESCRIPTION
       passwd changes passwords for user accounts. A normal user
       may only change the password for his/her own account, while
       the super user may change the password for any account.
       passwd also changes account information, such as the full
       name of the user, the user's login shell, or his/her
       password expiry date and interval.

   Password Changes
 Manual page passwd(1) line 1
```

که در شرح بالا، Synopsis اطلاعات مختصری در رابطه با نحوه کاربرد دستور است. بعلاوه تمام گزینه‌های موجود و اختیاری یا اجباری بودن آن‌ها را نیز نشان می‌دهد، اگر گزینه بین برکت‌ها باشد اختیاری است و در غیر این صورت، اجباری است.

[1] Shell
[2] Built-in

دستور man در ابتدا بخش‌های مختلفی را رابطه با سیستم‌عامل Unix داشت. اکنون نیز همان ساختار وجود دارد. بخش‌های مختلف man و نوع کمکی که از هر بخش می‌توان کسب نمود در جدول زیر آمده است. بنابراین شما به عنوان راهبر سیستم، معمولا به بخش‌های ۱، ۵ و نیز بخش ۸ نیاز خواهید داشت. برای اینکه اطمینان حاصل شود که تمام اطلاعات مربوط به یک دستور آورده شده است، از فرم man –a <command> استفاده می‌شود، که این نحوه استفاده، اطمینان خواهد داد که فقط بخش ۱ بررسی نمی‌شود، بلکه تمام بخش‌های man برای یافتن اطلاعاتی در رابطه با دستور مورد بررسی قرار خواهد گرفت و اگر می‌خواهید تنها در یک بخش خاص از man به دنبال شرح دستور باشید، آن را مانند زیر استفاده کنید:

man -5 passwd

یعنی فقط بخش ۵ را به دنبال شرح دستور مذکور بگردد. تا این مقطع، man بسیار مفید واقع خواهد شد اگر نام دستور را بدانیم، اما اگر نام دستور را ندانیم، سوییچی در man وجود دارد به نام k– که سبب یافتن دستوری متناسب با آنچه می‌خواهیم می‌شود. به دلیل لیست خروجی طولانی برای دستور man –k، می‌توان از دستور فیلتر کردن که در بخش Pipe and Redirection در اواخر همین پیش‌آگاهی آمده است، به شکل زیر برای جستجو در بخش خاصی مثلا بخش ۱، استفاده نمود.

Man –k time | grep 1

همچنین می‌توان از grep برای چاپ بخش‌هایی از فایل که مورد نظر باشد استفاده کرد. به عنوان مثال دستور

grep 10001 zipcodes.txt

در لینوکس، تنها سطوری را از فایل zipcodes.txt چاپ می‌نماید که رشته‌ی "10001" در آن‌ها باشد.

Section	Topic	Description
0	Header files	These are files that are typically in `/usr/include` and contain generic code that can be used by your programs.
1	Executable programs or shell commands	For the user, this is the most important section because it normally documents all commands that can be used.
2	System calls	As an administrator, you will not use this section on a frequent basis. The system calls are functions that are provided by the kernel. It's all very interesting if you are a kernel debugger, but normal administrators won't need this information.
3	Library calls	A library is a piece of shared code that can be used by several different programs. Typically, `man` pages that are documented in section 3 are relevant for programmers, not so much for Linux users and system administrators.
4	Special files	In here, the device files in the directory `/dev` are documented. These files are needed to access devices in a computer. This section can be useful for learning more about the workings of specific devices and how to address them using device files.
5	Configuration files	Here you'll find the proper format you can use for most configuration files on your server. If, for example, you want to know more about the way `/etc/passwd` is organized, use the entry for `passwd` in this section by issuing the command `man 5 passwd`.
6	Games	On a modern Linux system, this section contains hardly any information.
7	Miscellaneous	This section contains some information on macro packages used on your server.
8	System administration commands	This section does contain important information about the commands you will use on a frequent basis to change settings on your Linux machine.
9	Kernel routines	This is documentation that isn't even included as part of the standard install and optionally contains information about kernel routines.

۸-۳-۵-۲- استفاده از گزینه‌های help--

استفاده از این امکان ساده است. البته گرچه بسیاری از فرمان‌ها، چنین گزینه‌ای را مجاز می‌دانند، اما ممکن است همه آن‌ها با این گزینه کار نکنند. هدف اصلی آن، فراهم کردن شرحی کلی بر نحوه استفاده دستورات است که ممکن است خیلی طولانی‌تر باشند و در یک صفحه نمایش جا نشوند و بلافاصله صفحه نمایش را به سمت صفحات و جملات آخر هدایت کنند. در این‌صورت می‌توان از فیلتر less استفاده کرد. مانند زیر:

```
ls –help | less
```

۸-۳-٦- کار با متون

بخش مهمی از دستورات لینوکس در رابطه با کار با متون است. به عنوان مثال برای پیکربندی یک سرویس، باید پرونده متنی مربوط به آن را دستکاری کنید. همچنین نوشتن یک برنامه نیز مستلزم استفاده از یک پرونده متنی است.

گرچه امروزه ویرایشگرهای زیادی برای کار با لینوکس وجود دارد، Vi که حروف اول کلمه Visual است شناخته شده‌ترین و احتمالا پرکاربردترین آن‌هاست. امروزه ویرایش جدیدتر Vi که Vim نام دارد و معنی بهبود یافته یا Improve آن است، در اغلب لینوکس‌ها جایگزین شده است و کار با این دو شباهت‌های زیادی دارد. کار با ویرایشگر Vi برای کاربران ویندوز که به تازگی با لینوکس آشنا شده‌اند، دشوار است. ویرایشگر متن Vi از بسیاری از جنبه‌ها مهم است:

- به صورت پیش فرض بر همه توزیع‌های لینوکس موجود است در حالی که برخی ویرایشگرها مانند Emacs بر لینوکسی مانند Ubuntu نصب نیست.
- در شرایط بحرانی که سیستم هنوز بوت نشده، امکان کار با ویرایشگرهای پیشرفته مانند Emacs وجود ندارد.
- برخی دستورات رایج که راهبران لینوکس با آن‌ها زیاد کار می‌کنند، مبتنی بر همین ویرایشگر است، مثلا برای تغییر سطح دسترسی به دستور sudo، از دستور vosudo برای ویرایش این دستور استفاده می‌شود.

۸-۳-٦-۱- جزییات کار با Vi

یکی از دشواری‌های درک نحوه کار با Vi، دانستن حالات[1] اجرای آن است. Vi دو حالت اجرا دارد:

- حالت دستور[2]: برای وارد کردن دستورات
- حالت ورود[3]: برای وارد کردن متن

پیش از ورورد به حالت ورود، باید دستور مربوطه را صادر کرد. به عبارت دیگر در حالت دستور هستیم تا زمانی که دستوری برای ورود به حالت ورود صادر کنیم. دستورات حالت دستور حتی شامل تغییر مکان مکان‌نما[4] نیز هستند. البته Vi راه‌های متعددی فراروی استفاده کننده‌اش قرار می‌دهد. به عنوان مثال برای ورود به حالت ورود:

[1] Modes
[2] Command Mode
[3] Insert Mode
[4] Cursor

- فشردن i برای ورود متن در مکان کنونی مکان‌نما
- فشردن a برای الحاق متن پس از موقعیت کنونی مکان‌نما
- فشردن o برای گشودن سطری جدید، زیر موقعیت کنونی مکان‌نما
- فشردن 0 برای گشودن سطری جدید، بالای موقعیت کنونی مکان‌نما

برای ذخیره کردن یا خروج از محیط نیز بایستی به حالت دستور وارد شد و برای این امر باید Esc را بزنید. اگر Vi را بدون اسم پرونده صدا بزنید، در هنگام خروج باید اسم آن را برای ذخیره شدن بزنید.
برای خروج از محیط همراه با ذخیره پرونده، از گزینه wq: در حالت دستور استفاده کنید.
مکانیزم‌های cut, paste و copy نیز در این ویرایشگربرای کاربران جدید کمی غیر طبیعی است. در حالت دستوربا زدن کلید v می‌توان به انتخاب متن مورد نظر پرداخت و با زدن کلید d آن را cut نمود و به بافر منتقل نمود. کلید p نیز برای paste کردن استفاده می‌شود. برای حذف نیز از کلید delete می‌توان استفاده نمود اما دستورات اضافی دیگر را نیز می‌توان در حالت دستور اجرا نمود:

- استفاده از کلید x برای حذف تک کاراکتر
- استفاده از کلید dw برای حذف باقی متن. یعنی هر کاراکتری از وضعیت کنونی مکان‌نما تا انتهای کلمه
- استفاده از کلید dd برای حذف کامل سطر. این گزینه برای استفاده کاربران بسیار مفید واقع خواهد شد

در نهایت اینکه با دوبار زدن کلید g در حالت دستور، به ابتدای متن منتقل می‌شویم، در حالی که با دوبار زدن G مکان‌نما به انتهای متن منتقل می‌شود. خلاصه برخی دستورات برای مرور در ادامه آورده شده است، البته برای اختصار به برخی از آن‌ها در متن اشاره نشده است. در صورتی که فایلی را که تغییر کرده است، بخواهیم بدون مواجه شدن با پیام‌های خطا و یادآوری ببندیم، از علامت ! در انتهای دستور استفاده می‌کنیم، مثلا !q: را می‌توان بدین منظور به کار برد.

Command	Explanation
i	Opens insert mode for editing. Inserts text after the current cursor position.
Esc	Returns to command mode.
a	Opens insert mode for editing. Inserts text at the current cursor position.
o	Opens insert mode for editing. Opens a new line after the current line where the cursor is.
O	Opens insert mode for editing. Opens a new line before the current line where the cursor is.
:wq	Writes and quits the current document. Suppresses any warnings.
:w	Writes the current file using the same name. Appends a file name to write the file with another name.
:q	Quits without saving. Ignores any warnings.
u	Undoes the last command.
v	Enters visual mode to mark a block on which you can use commands.
d	Deletes the current selection.
y	Yanks (copies) the current selection.
p	Pastes.
g	Goes to the top of the current text file.
G	Goes to the bottom of the current text file.
/*text*	Searches *text* from the current position of the cursor downward.
?*text*	Searches *text* from the current position of the cursor upward.

۸-۳-۷- برنامه‌نویسی در لینوکس

بسیاری افراد فکر می‌کنند که برنامه‌نویسی در لینوکس یعنی استفاده از زبان C. درست است که لینوکس از ابتدا به زبان C نوشته شده است و عمده برنامه‌های کاربردی آن نیز به زبان C است، اما C تنها گزینه برنامه نویسی لینوکس نیست. سایر زبان‌هایی که با آن‌ها می‌توان تحت لینوکس برنامه نوشت به صورت خلاصه عبارتند از: Ada، C++، Eifell، Forth، Fortran، Icon، Java، JavaScript، Lisp، Perl، Prolog، Pascal، Python، PHP، Ruby، Smalltalk و ... است.

مثال:

برنامه زیر را در نظر بگیرید:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello World\n");
    exit(0);
}
```

سلسله مراتب اجرای آن طبق زیر است:

۱. پس از نوشتن آن در ویرایشگری مانند Vi، آن را با نام hello.c ذخیره می‌کنیم.

۲. با دستور زیر آن را compile و link کنید.

$ **gcc –o hello hello.c**

۳. و با دستور زیر آن را اجرا کنید.

$ **./hello**

Piping and Redirection -۱-۷-۳-۸

برای بهره‌برداری بیشتر از دستورات لینوکس می‌توان از Piping و Redirection استفاده کرد. با Piping می‌توان نتیجه‌ی یک دستور را به دستور دیگر هدایت کرد و با استفاده از Redirection می‌توان تعین کرد که خروجی دستور به کجا هدایت شود.

با Piping می‌توان دو یا بیشتر دستور را ترکیب کرد تا دستوری قوی‌تر با قابلیت‌های بیشتر فراهم نمود. استفاده‌ی صحیح از Piping، سبب ایجاد قابلیت‌های جالب و قوی بسیاری خواهد شد. برای یک راهبر حرفه‌ای، دستوری مانند زیر طبیعی است:

```
kill `ps aux | grep y2 | grep -v grep | awk '{ print $2 }'`
```

به عنوان مثال، زمانی که خروجی یک دستور بزرگتر از صفحه نمایش است، می‌توان از دستور مفیدی به نام less برای رویت صفحه به صفحه آن استفاده کرد.

مثلا دستور زیر را ببینید:

ls –r / |less

دستور بسیار مفید دیگر، عملگر < یا > است، که با نام redirection نام برده می‌شود. خروجی دستورات خط فرمان، به صورت پیش‌فرض بر صفحه نمایش ظاهر می‌شود اما می‌توانید در لینوکس، این حاصل را به یک خروجی دیگر مانند یک پرونده هدایت کنید و همانطور که در Shell Scripting در ویندوز نیز رایج است، از < برای انتقال خروجی به یک پرونده و ایجاد همزمان آن استفاده می‌شود در حالی که از >> برای

الحاق نمودن خروجی کنونی به پرونده‌ای که هم‌اکنون وجود دارد استفاده می‌شود. اگر خروجی یک دستور همراه با خطا باشد، می‌توان از 2> برای هدایت خطا به یک پرونده‌ی خاص استفاده کرد. البته می‌توان به جای پرونده، خروجی یک دستور را به دستگاه نیز فرستاد.

۸-٤- مراجع

1. Neil Matthew and Richard Stones, **Beginning Linux Programming**, 4'th Ed., Wiley Publishing, Inc., 2007.
2. Keir Thomas, **Beginning Ubuntu Linux From Novice to Professional**, Apress, 2006.
3. Sander van Vugt, **Beginning the Linux Command Line**, Apress, 2009.
4. Paul G. Sery, **Ubuntu Linux For Dummies**, John Wiley & Sons, 2007.
5. Matthias Kalle Dalheimer, Terry Dawson, Lar Kaufman and Matt Welsh, **Running Linux**, 4'th Ed., O'Reilly, 2002.

۶. صدیقی مشکنانی، محسن. **دستورکار آزمایشگاه سیستم‌عامل**. دانشکده برق و کامپیوتر دانشگاه صنعتی اصفهان. زمستان ۱۳۷۷.

۸-۵- دستور کار

سیستم را روشن کنید و پس از راه‌اندازی Linux، وارد حساب کاربری خود شوید. برای این کار نام کاربری و کلمه‌ی عبور خود را از مربی آزمایشگاه دریافت کنید. از Application به Accessories بروید و گزینه Terminal را انتخاب کنید تا اعلان سیستم‌عامل در خط فرمان را مشاهده کنید. سپس گام‌های زیر را انجام دهید:

۱. فرمان ls، دارای گزینه‌هایی است که برخی از آن‌ها عبارتند از: a-، l-. از راهنمای سیستم عملکرد دو گزینه‌ی فوق و سه گزینه‌ی دلخواه دیگر این دستور را بیابید و در کاربرگ خود یادداشت کنید.

۲. با فرمان mkdir یک شاخه‌ی جدید ایجاد کنید. آیا توسط این فرمان می‌توان شاخه و زیرشاخه‌ی جدیدی را همزمان ایجاد کرد؟ چگونه؟

۳. با استفاده از دستور cp سعی کنید از یک شاخه و تمام محتویات آن کپی تهیه کنید. چگونگی انجام این کار را شرح دهید. در صورت بروز مشکل به راهنمای سیستم مراجعه کنید.

۴. فرمان rmdir را امتحان کنید. آیا در حذف فهرست توسط این فرمان محدودیتی وجود دارد؟ توضیح دهید.

۵. به راهنمای سیستم فرمان man مراجعه کنید و سه گزینه از گزینه‌های دستور man را مراجعه کرده و توضیحات مختصری در مورد آن‌ها بنویسید. سپس هر یک از آن‌ها را آزمایش کنید.

۶. با فرمان ls شاخه‌ها و محتویات داخل آن‌ها را مشاهده کرده و از میان آن‌ها پرونده‌ای بیابید که اجازه‌ی خواندن آن را نداشته باشید. سپس سعی کنید با فرمان‌هایی که می‌دانید آن را بخوانید. با چه پیامی مواجه می‌شوید؟ این کار را برای پرونده‌هایی که اجازه‌ی نوشتن آن را ندارید، تکرار کنید. نتایج به دست آمده را در کاربرگ خود بنویسید.

۷. الف) دقیقا بنویسید که دستور زیر چه کاری انجام می‌دهد؟ (محدوده‌ی عمل دستور هم ذکر شود)

```
$ find . -name '*.jpg'
```

ب) پرونده hosts را در کل سیستم جستجو کنید. هم دستوری را که برای یافتن آن به کار می‌برید بنویسید و هم در صورت یافتن پرونده مذکور، مسیر یا مسیرهای آن را بنویسید.

۸. از راهنمای سیستم، نام پنج فراخوان سیستمی را بیابید و در کاربرگ‌های خود بنویسید. کار یکی از آن‌ها را توضیح دهید.

۹. وارد برنامه Vi شوید و یک برنامه‌ی کوچک به زبان C، همانند برنامه موجود در پیش‌آگاهی بنویسید که پیام‌هایی را بر روی صفحه نمایش چاپ کند. آن را در پرونده‌ای به نام first.c ذخیره کنید. برنامه را ترجمه و اجرا کنید. نام فایل اجرایی را firstexe بگذارید. حال دستورات زیر را اجرا کنید و حدس بزنید که دستور file چه کاری انجام می‌دهد. حدس خود را بنویسید.

```
$ file first.c
$ file firstexe
```

۱۰. اجازه‌های دسترسی به فایل اجرایی آزمایش ۹ را طوری تنظیم کنید که هیچکس حتی مالک آن نیز امکان اجرا را نداشته باشد.

الف) دستور مربوط برای این امر را بنویسید.

ب) حال دستور اجرای برنامه first را بنویسید و پیام ظاهر شده را در کاربرگ یادداشت کنید.

۱۱. فرمان ps را اجرا کنید. خروجی این فرمان را ببینید و درباره اطلاعاتی که توسط این فرمان در مورد فرایندها نشان داده می‌شود توضیح دهید.

۱۲. برنامه‌ی first یا ویرایشگر Viرا مجددا اجرا کنید. سپس روی پایانه‌ی چهار بروید و دوباره وارد حساب خود شوید. فرمان ps را اجرا کنید. نتیجه‌ی اجرای این فرمان با نتیجه مشاهده شده در گام قبلی چه تفاوتی دارد؟ در رابطه با محدوده‌ی این دستور چه نتیجه‌ای می‌توان گرفته؟ توضیح دهید.

۱۳. برنامه‌ی sample.c زیر را نوشته واجرا کنید و خروجی آن را مشاهده نمایید. سپس در اجرای بعد، فایلی متنی با نام file.txt با محتوی سطر زیر را به عنوان ورودی به آن بدهید (با کمک Redirection). قالب دستور مربوط به redirect را در کاربرگ یادداشت کنید.

Sample.c

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
int main()
{
        int ch;
        while((ch = getchar()) != EOF) {
                putchar(toupper(ch));
        }
exit(0);
}
```

file.txt

This is a file, file.txt, it is all lower case.

۱۴. همانطور که در پایانه چهار قرار دارید، فرمان top را وارد کنید و در مورد آنچه بر صفحه نمایش می‌بینید، توضیح دهید.

۱۵. همانطور که در پایانه چهار قرار دارید، کاربری به نام user2 و رمز عبور ۱۲۳ ایجاد کنید و به پایانه پنج بروید و تحت کاربر جدید Log in کنید و دستور who را اجرا کنید.

الف) چه اطلاعاتی به شما داده می‌شود؟

ب) به نظر شما برای بازگشت به حالت گرافیکی چه باید کرد؟

آزمایش هشتم

آشنایی با مدیریت فرایندها در لینوکس

۹-۱- هدف

آشنایی با:

- فرایند[1] و حالات مختلف آن
- انواع روش‌های ایجاد فرایند جدید در لینوکس و خاتمه آن
- اجرای فرایندها در پس‌زمینه

۹-۲- پیش‌آگاهی

درک عملی فرایندها در لینوکس، اضافه بر اینکه به مفهومی مهم در سیستم‌عامل عینیت می‌بخشد، سبب راهبری بهتر و بیشتر بر سیستم‌عامل لینوکس خواهد شد و به عنوان مثال در مواقعی مانند از کارافتادن برنامه، یا حتی تغییر اولویت فرایندها می‌توان از آن بهره برد. به ازای اجرای هر برنامه ممکن است یک یا بیشتر فرایند فعال شود.

[1] Process

۹-۲-۱- فرایندها در لینوکس

در زمان روشن شدن سیستم، ابتدا هسته[1] شروع به کار می‌کند. هسته، متولی آغاز اولین فرایند است که عبارتست از فرایند init. این فرآیند مسؤل تمام فرایندهای دیگر است. این فرایند فرایند دیگر را به عنوان فرایند فرزند[2] اجرا خواهد کرد. برای مثال، mingetty از init شروع می‌شود، که مسؤل گشودن یک پوسته ورود[3] است. از mingetty، فرایند bash آغاز به کار می‌کند که اجازه کار با خط فرمان را برای کاربران لینوکس فراهم می‌کند. بنابراین یک رابطه پدر و فرزندی بین فرایندهای لینوکس وجود دارد. Init اولین فرایند است که mingetty از آن مشتق می‌شود سپس bash به عنوان فرزند از mingetty به وجود می‌آید و زین‌پس هر دستوری که در خط فرمان داده شود به عنوان فرزندی از bash به حساب می‌آید.

لینوکس یک صف برای فرایندهای آماده به اجرا نگه می‌دارد. لینوکس به‌طور پیش‌فرض به هریک از فرایندها بازه‌های زمانی مساوی اختصاص می‌دهد، اما اگر فرایندی نیاز به میزان بیشتری از زمان نسبت به سایر فرایندها داشته باشد، بایستی این کار را با فراخوانی تابعی به نام nice برای افزایش یا حتی کاهش میزان زمان مجاز برای یک فرایند فراخوانی کرد.

گاهی ممکن است نیاز باشد که یک فرایند را متوقف کنیم. به عنوان مثال ممکن است یک فرایند دیگر به هیچ سیگنالی پاسخ ندهد و نتوان با آن کار کرد یا این‌که به گونه‌ای رفتار کند که به سایر فرایندها آسیب برساند. در این صورت هسته‌ی سیستم عامل به فرایند پدر مربوطه خواهد گفت که لازم است فرزند آن را متوقف کند. در شرایط معمول، فرایند پدر تا زمان حیات فرایند فرزند، در سیستم وجود دارد، اما در شرایط غیرمعمول ممکن است فرایند فرزند در حالی در سیستم باقی مانده باشد که پدر متوقف شده باشد، در این حالت نمی‌توان فرایند فرزند را متوقف کرد و گوییم به حالت Zombie یا جادویی رفته است. بدین ترتیب دیگر نمی‌توان از خط فرمان فرایند Zombie را متوقف کرد و تنها راه، شروع مجدد[4] سیستم است.

اغلب ظهور فرایند Zombie، سایر فرایندها را نیز تحت تأثیر قرار می‌دهد. علت چنین رویدادی نیز معمولا به اشتباهات برنامه نویسی برمی‌گردد که چنین فرایندهایی را ایجاد می‌کند. سایر حالات دیگری که فرایند می‌تواند در آن‌ها قرار گیرد را می‌توان با دستور ps aux مشاهده کرد.

[1] Kernel
[2] Child
[3] Login shell
[4] Restart

۹-۲-۲- حالات مختلف فرآیند

همانطور که در آزمایش پیش مشاهده شد، دستور ps برای مشاهده وضعیت فرایندها استفاده می‌شود. یکی از مشخصه‌هایی که دستور ps –al برای یک فرایند نشان می‌دهد، Stat نام دارد که حالت فرایند را به صورت یک کد سه حرفی نشان می‌دهد. حرف اول این کد یکی از حروف زیر است:

- R: برای فرایندهای در حال اجرا (Running)
- S: برای فرایندهای در منتظر (Waited)
- Z: برای فرایندهای جادویی (Zombie)
- T: برای فرایندهایی که متوقف (stopped) یا ردگیری (Trace) شده‌اند (یا به اصطلاح آماده‌ی اجرا هستند)
- D: برای فرایندهایی که در حالت انتظار بدون بازپس‌گیری[1] یا انتظار در حافظه‌ی جانبی هستند.

اگر حرف دوم این کد W باشد، نشان‌دهنده‌ی آن است که فرایند مورد نظر هیچ صفحه‌ای در حافظه‌ی اصلی ندارد و به‌طور کامل مبادله (Swap) شده است.

حرف سوم در مورد زمانبندی و اولویت فرایندهاست. اگر این حرف N باشد، نشان دهنده‌ی این است که ارزش مطلوب (Nice Value) برای فرایند بیشتر از صفر است (این نوع فرایند اولویت بیشتری دارد و وقت بیشتری را می‌تواند از پردازنده بگیرد).

در سیستم‌عامل لینوکس، یک فرایند می‌تواند ۹ حالت مختلف داشته باشد. این حالات در شکل و جدول زیر آمده‌اند. لازم به ذکر است که منابع مختلف، با دیدگاه‌های خلاصه و تفصیلی به حالات فرایند در لینوکس، تعدادهای کمتر و بیشتر هم برای حالات فرایند بیان کرده‌اند.

جدول ۹-۱- حالات مختلف فرایندها

ردیف	نام	شرح
۱	اجرا در فضای کاربر	فرایند موقع اجرای عملیات عادی خود در این حالت قرار دارد
۲	اجرا در فضای هسته	فرایند، هنگام استفاده از سرویس‌های هسته در این حالت قرار دارد. مانند اجرای فراخوانی سیستم و روال خدماتی وقفه‌ها
۳	آماده به اجرا در حافظه‌ی اصلی	فرایند در حال اجرا نیست ولی در حافظه‌ی اصلی قرار دارد و منتظر اختصاص وقت پردازنده است تا اجرا شود

[1] Preemption

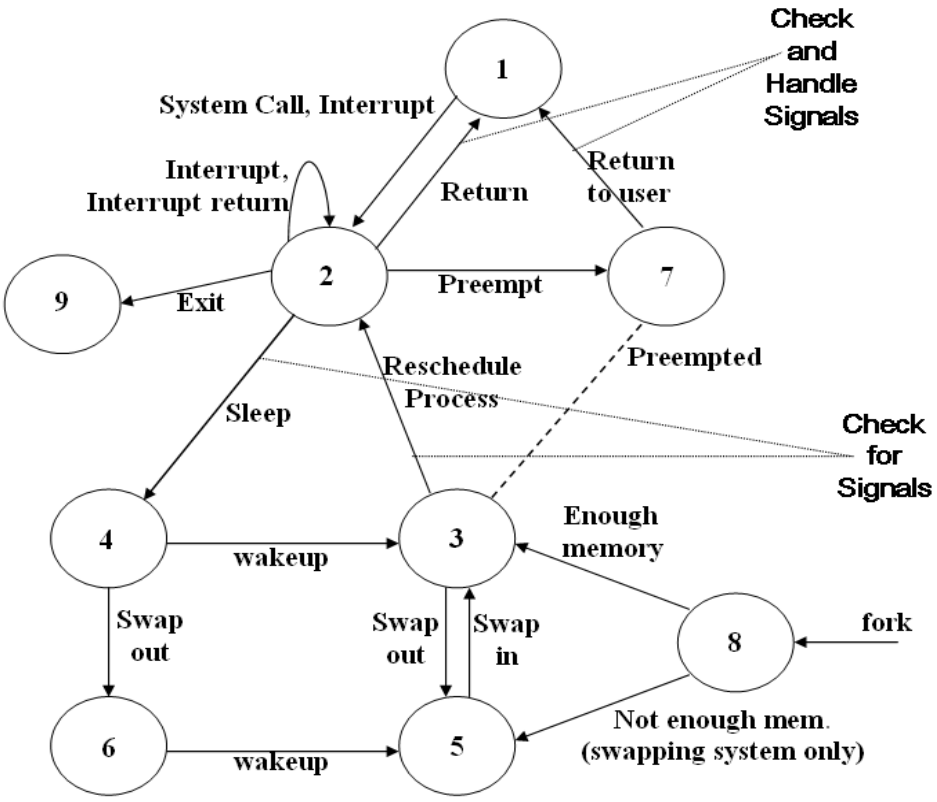
۴	متوقف (Sleep) در حافظه‌ی اصلی	فرایند در حافظه‌ی اصلی است و منتظر پایان یافتن عملی مانند ورودی- خروجی است تا دنباله‌ی اجرایش را از سربگیرد
۵	آماده به اجرا در حافظه‌ی جنبی	فرایند آماده‌ی اجراست ولی به دلیل کمبود حافظه، به حافظه‌ی جانبی منتقل شده است و قبل از آن‌که وقت پردازنده به آن تخصیص داده شود، باید به حافظه‌ی اصلی منتقل شود
۶	متوقف در حافظه‌ی جنبی	مشابه به حالت ۴ است اما فرایند به علت کمبود حافظه، توسط هسته به حافظه‌ی جانبی منتقل شده است تا حافظه‌ی کافی برای فرایندهای در حال اجرا فراهم گردد
۷	قبضه‌شده (Preempted)	فرایند از حالت اجرا در فضای کاربر به حالت اجرا در فضای هسته منتقل می‌گردد، ولی در این هنگام زمان اختصاص یافته به آن پایان می‌یابد و زمانبند فرایندها، فرایند دیگری را برای اجرا انتخاب می‌کند
۸	ایجاد	هر فرایندی که ایجاد می‌شود، نخست در این حالت قرار می‌گیرد، در این حالت نه در حال اجراست و نه در حال توقف. این حالت نقطه‌ی شروع تمام فرایندها (به‌جز فرایند با pid صفر) است
۹	جادویی (Zombie)	هر فرایند توسط تابع exit، با به‌جا گذاشتن اطلاعاتی برای پدر خویش (فرایندی که آن را ایجاد کرده است)، به کار خویش خاتمه می‌دهد. هسته، تمام منابع، به‌جز کد خروج فرایند را از این نوع فرایندها می‌گیرد. این منابع شامل حافظه، پردازنده و ... است. اگر فرایندی، فرایند فرزندی را ایجاد کند، مسؤلیت اتمام آن را نیز برعهده دارد. بنابراین در صورتی که فرایند فرزند، زودتر خاتمه یابد، کماکان در سیستم وجود خواهد داشت (حالت جادویی) تا اینکه یا فرایند پدر با استفاده از exit code فرایند فرزند، در یک فراخوانی سیستمی به نام wait یا waitpid، آن را مختومه کند یا اینکه فرایند پدر خود به صورت خودکار خاتمه یابد و تمام منابع اختصاصی‌اش آزاد شود. این فراخوانی‌های سیستمی باعث می‌شوند، پدر از وضعیت (کد) خروج فرزندش مطلع شود. اگر پدر منتظر دریافت کد خروج فرزندش نیز نشود و کارش را به پایان برساند در حالی‌که هنوز حیات فرزند ادامه دارد، فرزند بی‌سرپرست (orphan) می‌شود. مدیریت این نوع فرایندها به عهده‌ی سیستم‌عامل است.

وقتی فرایند پدر با فراخوان سیستم[1] fork، فرایند جدیدی ایجاد می‌کند، فرایند ایجاد شده، بسته به مقدار حافظه‌ی آزاد سیستم، وارد یکی از مراحل ۳ یا ۵ خواهد شد. برای سادگی فرض کنید فرایند وارد مرحله‌ی ۳ می‌گردد. وقتی زمانبند فرایندها به صورت تصادفی آن‌را برای اجرا برگزید، فرایند وارد مرحله‌ی اجرا در

[1] فراخوان سیستم آن دسته از توابعی هستند که از آن‌ها برای استفاده از امکانات هسته‌ی سیستم‌عامل فراخوانی می‌کنیم. کد این برنامه‌ها در داخل هسته قرار دارد و موقع استفاده از آن‌ها، کدشان ضمیمه برنامه نمی‌شود.

فضای هسته گشته و در این هنگام کار فراخوان fork به اتمام می‌رسد. پس از این مرحله ممکن است فرایند به مرحله‌ی ۱ برود.

بعد از یک برش زمانی، به علت وقوع وقفه‌ی ساعت سیستم، فرایند به مرحله‌ی ۲ باز می‌گردد (چون تمام فرایندها این وقفه را دریافت می‌کنند). هسته از این وقفه‌ی ساعت برای زمانبندی و اولویت‌بندی فرایندها استفاده می‌کند. پس از این زمانبندی، ممکن است فرایند دیگری برای اجرا انتخاب گردد. در این هنگام فرایند قبلی وارد حالت ۷ می‌گردد و منتظر می‌ماند تا برای اجرای دوباره انتخاب گردد. اگر فرایند در حال اجرا در فضای کاربر، بخواهد از سرویس‌های هسته استفاده کند، وارد مرحله‌ی ۲ می‌گردد. اگر این سرویس، کاری مانند ورودی- خروجی باشد، فرایند وارد مرحله‌ی ۴ می‌شود و خود را متوقف می‌سازد. این توقف تا زمانی ادامه می‌یابد که به واسطه‌ی اتمام کار ورودی- خروجی، وقفه‌ای به پردازنده برسد و سرویس روتین وقفه، فرایند مربوطه را بیدار کند. این کار باعث می‌شود تا فرایند به مرحله‌ی ۳ باز گردد و منتظر شود تا هسته آن‌را زمانبندی کند.



جدول ۹-۲ - نمودار حالات مختلف فرایندها بر اساس شماره ردیف‌های جدول ۱

اگر فضای کافی برای فرایندهای در حال اجرا، در حافظه وجود نداشته باشد، فرایند مبادله کننده[1] (با شماره‌ی صفر) بعضی از فرایندها را بر روی دیسک منتقل می‌کند و به این ترتیب فرایند را وارد مرحله‌ی ۵ می‌کند. همین اتفاق ممکن است برای فرایندهای متوقف در حافظه (حالت ۴) نیز اتفاق بیفتد. وقتی کار فرایند به پایان رسید، وارد مرحله‌ی ۹ می‌شود.

۹-۲-۳- ایجاد فرایند جدید

برای ایجاد یک فرایند جدید، از یک فراخوان سیستم به نام fork استفاده می‌شود. خروجی fork از جنس pid_t (در حقیقت یک عدد صحیح) است. این خروجی همان شناسه‌ی فرایند (شماره‌ی فرایند PID) است. با احضار فراخوان سیستم fork، فرایند جدیدی ایجاد می‌شود که پدر آن، فرایندی است که fork را فراخوانده است. فرایند جدید در کنار بقیه‌ی فرایندهای موجود در سیستم قرار می‌گیرد و منتظر می‌شود تا وقت پردازنده به او اختصاص یابد. اما در این فرایند چه چیزی اجرا می‌شود؟ پاسخ این است که کد اجرایی این فرایند دقیقاً همانند پدرش است و همان چیزی را اجرا می‌کند که پدرش اجرا می‌کرده است، اما از نقطه‌ی فراخوانی به بعد. به عبارت دیگر از نقطه‌ی فراخوانی به بعد، کد پدر در دو خط موازی اجرا می‌شود، یکی توسط پدر و دیگری توسط فرایند فرزند که از آنجا به بعد با پدر یکسان است. فرایند فرزند، محیط متغییرها، پرونده‌های باز، شناسه‌های پرونده‌ی پدر را به ارث می‌برد. البته تفاوت‌های اندکی نیز بین این دو فرایند وجود دارد؛ از جمله شماره‌ی فرایند (PID) و شماره‌ی فرایند پدر (PPID).

قطعه کد زیر را در نظر بگیرید:

```
Pid_t child_id;
child_id = fork();
printf("forked\n");
```

با اجرای این دستور، یک فرایند جدید ایجاد خواهد شد که دارای کدی به صورت فوق بوده، شمارنده‌ی برنامه‌ی[2] آن به دستورالعمل بعد از fork اشاره می‌کند. اکنون دو فرایند وجود دارند که دستورالعمل بعدی را اجرا خواهند کرد و در نتیجه کلمه‌ی fork دو بار چاپ خواهد شد. این یک روش چند نخی در لینوکس است که مدت‌ها پیش از سیستم عامل ویندوز، آن را در اختیار برنامه‌نویسان قرار داده بود.

نکته: pid_t یکی از انواع تایپ‌های صحیح زبان C تحت لینوکس است که محتوی آن می‌تواند شماره‌ی یک فرایند باشد. شماره‌ی فرایند (PID) مشخصه‌ای است که سیستم عامل توسط آن یک فرایند را می‌شناسد و به آن دسترسی پیدا می‌کند.

[1] Swapper
[2] Program Counter

۹-۲-٤- اجرای عملیاتی در فرایند فرزند متفاوت با عملیات فرایند اصلی

تابع system: می‌توان با کمک این تابع کتابخانه‌ای[1] برنامه‌ای را از داخل برنامه‌ی دیگر فراخوانی کرد و به این ترتیب، سبب ایجاد فرایند جدیدی مستقل از فرایند کنونی شد. شکل کلی آن به صورت زیر است:

```
int system(const char *string)
```

این تابع، دستوری را که در قالب رشته به آن داده شده است، با فراخوانی پوسته انجام می‌دهد، درست همانند اجرای یک دستور معمولی. البته بعد از انجام این تابع، کنترل اجرا به برنامه‌ی فراخوان برخواهد گشت.

توابع exec: فراخوان‌های exec مجموعه توابع مفیدی هستند که اجرای یک دستور سیستم عامل را (که قاعدتا از خط فرمان اجرا می‌شود) در داخل برنامه امکان‌پذیر می‌سازند. یک تابع exec، فرایند جاری را با فرایندی جدید که در آرگومان‌های path و file مشخص شده است، جایگزین می‌کند. از این امر می‌توان برای تغییر اجرای یک فرایند به فرایند دیگر استفاده کرد. به عنوان مثال، در یک برنامه ممکن است ابتدا کاربر را شناسایی کنیم و سپس به او اجازه‌ی اجرای برنامه‌ی دیگری را بدهیم. بدین ترتیب عملکرد این توابع در لینوکس همانند عملکردشان در سیستم‌عامل قدیمی DOS است؛ با این تفاوت که در DOS، اجرای exec شبیه فراخوانی یک روال معمولی است و پس از اتمام کار، به بعد از محل فراخوانی exec برمی‌گردیم. اما در لینوکس، تمام شدن اجرای exec، به اجرای برنامه (که شامل تابع main در فراخوانی exec است) خاتمه خواهد داد. توابع مختلف exec که می‌توان از آن‌ها استفاده کرد، عبارتند از:

```
#include <unistd.h>

int execl(const char *path, const char *arg0, ..., (char *)0);
int execlp(const char *file, const char *arg0, ..., (char *)0);
int execle(const char *path, const char *arg0, ..., (char *)0, char *const envp[]);
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
int execve(const char *path, char *const argv[], char *const envp[]);
```

برای اطلاعات بیشتر در مورد این توابع، می‌توان به کتاب‌های آموزشی زبان C مراجعه کرد، اما برای آشنایی بیشتر، به مثال زیر توجه کنید:

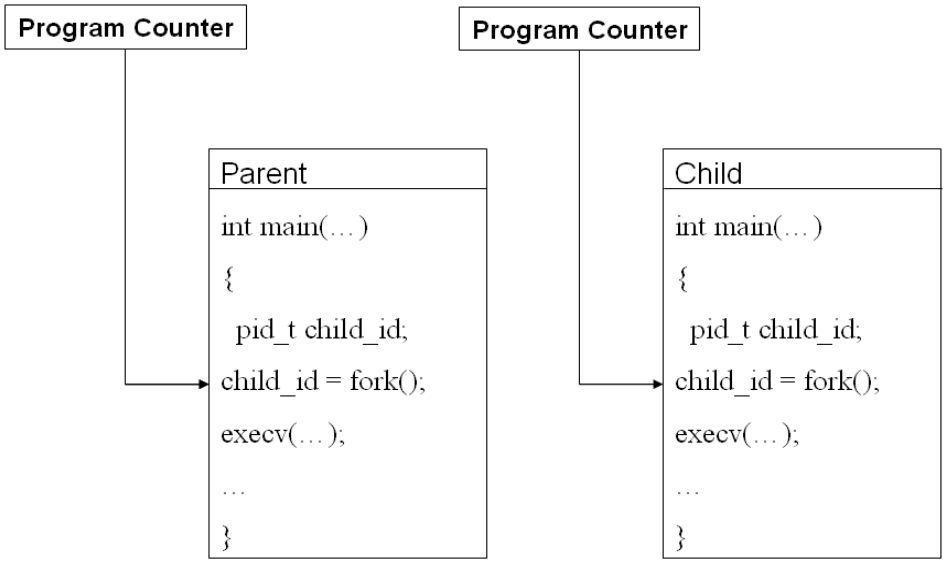
[1] Library function

```
/* Example of an argument list */
/* Note that we need a program name for argv[0] */
char *const ps_argv[] = {“ps”, “ax”, 0};
/* Example environment, not terribly useful */
char *const ps_envp[] = {“PATH=/bin:/usr/bin”, “TERM=console”, 0};
/* Possible calls to exec functions */
execl(“/bin/ps”, “ps”, “ax”, 0); /* assumes ps is in /bin */
execlp(“ps”, “ps”, “ax”, 0); /* assumes /bin is in PATH */
execle(“/bin/ps”, “ps”, “ax”, 0, ps_envp); /* passes own environment */
execv(“/bin/ps”, ps_argv);
execvp(“ps”, ps_argv);
execve(“/bin/ps”, ps_argv, ps_envp);
```

در اینجا به نظر می‌رسد اگر بتوان تابع exec را در مسیری جدا از مسیربرنامه‌ی اصلی اجرا کرد، مشکل اجرای فرایندی متفاوت با فرایند پدر در فرایند فرزند، قابل حل خواهد بود. اگر برای ایجاد این مسیر جدید، بعد از fork یک فراخوانی exec قرار دهیم (مطابق شکل ۲)، این تابع در فرایند پدر نیز اجرا خواهد شد که مطلوب نیست (مانند دستور printf در قطعه کد بالا داشتیم).

نکته رفع مشکل فوق توجه به تفاوت بین مقدار child_id در پدر و فرزند ایجاد شده توسط دستور fork است. زیرا عبارت child_id=fork()، در صورت ایجاد موفقیت‌آمیز فرایند فرزند، در هر دو فرایند پدر و فرزند مقدار خواهد گرفت. در فرایند پدر مقداری بزرگتر از صفر (به عنوان شماره فرزند ایجاد شده) در child_id قرار خواهد گرفت و در فرایند فرزند، مقدار صفر در این متغییر ذخیره خواهد شد.

اگر اشکالی در ایجاد فرایند فرزند به وجود آید، fork مقدار ۱- را برخواهد گرداند که به این ترتیب چون فرایند فرزند ایجاد نشده در child_id از فرایند پدر ذخیره خواهد شد.



شکل ۹-۱- وضعیت فرایند اصلی (پدر) و فرایند ایجاد شده (فرزند) بلافاصله بعد از fork

با توجه با مطالب بالا می‌توان بعد از fork، قسمت‌های اختصاصی پدر و فرزند را به شکل زیر مشخص کرد:

```
child_id = fork();
if (child_id = = -1) {
        /* خطا در ایجاد فرایند فرزند:
        معمولا در این‌حالت، تابع با یک کد خطا ترک می‌شود. */
}
if (child_id = = 0) {
        /* قسمت اختصاصی فرزند:
        کارهای مربوط به فرایند فرزند در این قسمت انجام می‌شود.
        این بخش با اینکه در فرایند پدر نیز وجود دارد،
        هرگز در فرایند پدر اجرا نمی‌شود. به این علت که در صورت موفقیت
        fork، child_id در فرایند پدر، مقـــدار بزرگتر از صفر به
                                خود می‌گیرد.
        */
}
else {
        /* قسمت اختصاصی پدر:
        کارهای اختصاصی فرایند پدر نیز در این قسمت انجام می‌گیرد. زیرا
        در صورت موفقیت fork، child_id در فرایند فرزند مخالف صفر
                                خواهد شد.
        */
```

قسمت‌هایی که مقدار برگشتی از fork، ۱- نبوده است، می‌تواند به عنوان قسمت اشتراکی بین هر دو فرایند نیز در نظر گرفته شود. البته مانند کد بالا می‌توان فرایندهای پدر و فرزند را از هم جدا کرد. یک روش دیگر برای نوشتن کد بالا، مانند زیر است:

```
child_pid = fork ();
switch (child_pid)
{
        case -1:        /* خطا در ایجاد فرایند */
        case 0:         /* فرایند فرزند */
        default:        /* فرایند پدر */
}
```

۹-۲-۵- متوقف کردن یک فرایند

اگر فرایندی با استفاده از fork، فرایند جدیدی را به وجود آورد. به محض ایجاد فرایند فرزند، دو فرایند پدر و فرزند به موازات هم اجرا خواهند شد. ولی گاهی این عمل مد نظر نیست، بلکه مثلا فرایند پدر باید منتظر فرایند فرزند بماند تا کار او تمام شود، آنگاه به کار خود ادامه دهد. برای این‌کار فرایند پدر باید از فراخوان سیستم wait استفاده کند. این تابع به صورت زیر تعریف شده است:

```
int wait (int* status)
```

فرایندی که این تابع را احضار کرده باشد، تا اتمام کار یکی از فرزندانش صبر خواهد کرد (مقدار خروجی این تابع همان PID فرایند فرزندی است که کارش تمام شده است و کد خروج این فرزند در پارامتر status ظاهر خواهد شد). اگر فرایند احضار کننده، هیچ فرزندی نداشته باشد، این تابع را بلافاصله بازگشت می‌کند.

فراخوان سودمند دیگر برای متوقف کردن یک فرایند، waitpid است. در مورد آن می‌توان از صفحات راهنمای موجود در لینوکس کمک گرفت. این تابع، فرایند جاری را قادر می‌سازد تا منتظر اتمام کار فرایند فرزندی شود که شماره‌ی آن فرایند فرزند را به عنوان پارامتر به تابع waitpid ارائه کرده‌ایم.

۹-۲-۶- اجرای فرایندها در پس‌زمینه[1]

با افزودن علامت & به آخر هر فرمان یا برنامه، می‌توان آن را در پس‌زمینه اجرا کرد. در این حالت، لینوکس علاوه بر اجرای برنامه‌ی ذکر شده، با دادن اعلان خط فرمان، آمادگی خود را جهت گرفتن

[1] Background

فرمان، یا برنامه‌ای دیگر اعلام می‌کند (بدون اینکه اجرای برنامه‌ی قبلی پایان یافته باشد). بدین ترتیب دو یا چند فرمان را می‌توان با هم اجرا کرد و شیوه‌ای دیگر از چندوظیفه‌ای را پیاده‌سازی نمود.

مثال:

```
ls –l &
```

۹-۳- مراجع

7. Neil Matthew and Richard Stones, **Beginning Linux Programming**, 4'th Ed., Wiley Publishing, Inc., 2007.
8. Keir Thomas, **Beginning Ubuntu Linux From Novice to Professional**, Apress, 2006.
9. Sander van Vugt, **Beginning the Linux Command Line**, Apress, 2009.
10. Matthias Kalle Dalheimer, Terry Dawson, Lar Kaufman and Matt Welsh, **Running Linux**, 4'th Ed., O'Reilly, 2002.

۱۱. صدیقی مشکنانی، محسن. **دستورکار آزمایشگاه سیستم‌عامل**. دانشکده برق و کامپیوتر دانشگاه صنعتی اصفهان. زمستان ۱۳۷۷.

۹-٤- دستور کار

توجه: برای افزایش سرعت نوشتن برنامه‌ها، می‌توانید از Text Editor موجود در Accessories، واقع در Application استفاده کنید.

سیستم را روشن کنید و پس از راه‌اندازی Linux، وارد حساب کاربری خود شوید. برای این کار نام کاربری و کلمه‌ی عبور خود را از مربی آزمایشگاه دریافت کنید. از Application به Accessories بروید و گزینه Terminal را انتخاب کنید تا اعلان سیستم‌عامل در خط فرمان را مشاهده کنید. سپس آزمایش‌های زیر را انجام دهید:

۱. دستور pstree را اجرا نموده و علاوه برذکر عنصر ریشه‌ی آن، دو مورد آشنای داخلی را نیز یادداشت نمایید.

۲. برنامه‌ای مانند ماشین حساب را باز کنید و با دستور kill به اجرای آن خاتمه دهید. نحوه یافتن آن برنامه و نحوه استفاده از دستور kill را در کاربرگ بنویسید.

۳. برنامه زیر را در فایلی به نام first.c ذخیره و اجرا کنید. خروجی را در کاربرگ خود یادداشت کنید. آیا انتظار می‌رود با اجرای مجدد آن، نتایج متفاوتی حاصل شود؟ چرا؟

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        pid_t pid;
        char *message;
        int n;

        printf("fork program starting\n");
        pid = fork();
        switch(pid)
        {
        case -1:
                perror("fork failed");
                exit(1);
        case 0:
```

```
                message = “This is the child”;
                n = 5;
                break;
        default:
                message = “This is the parent”;
                n = 3;
                break;
        }

        for(; n > 0; n--) {
                puts(message);
                sleep(1);
        }
exit(0);
}
```

۴. قطعه برنامه‌ی زیر را با نام execp.c ذخیره کنید و اجرا نمایید و بیان کنید که آیا خروجی دستور printf(“Done.\n”); نیز قابل رویت است؟ چرا؟

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        printf(“Running ps with execlp\n”);
        execlp(“ps”, “ps”, “ax”, 0);
        printf(“Done.\n”);
        exit(0);
}
```

۵. با استفاده از تابع system، سطر execlp(“ps”, “ps”, “ax”, 0); را از تمرین قبل جایگزین کرده و خروجی را مقایسه کنید و علاوه بر دستور system مورد استفاده، تفاوت خروجی را همراه با دلیل آن شرح دهید.

۶. با استفاده از گزینه‌ی & برای فرستادن برنامه به پس‌زمینه و با استفاده از دستور sleep(i) که در آزمایش ۲ فراگرفتید و برای ایجاد درنگ در برنامه به مدت i ثانیه است، دو برنامه‌ی ساده به نام‌های 11.c و 12.c بنویسید که اولی خروجی First Program Finished را بعد از بیست ثانیه، ظاهر کند و دومی به مدت بیست ثانیه در هر یک ثانیه، پیام Second Program running را چاپ نماید. ابتدا هر دو را کامپایل کنید، سپس برنامه‌ی 11 را به اجرا در پس‌زمینه بفرستید، سپس

برنامه‌ی 12 را اجرا کنید و در نهایت خروجی را به مدرس نشان دهید. متن برنامه‌ها را در کاربرگ قید کنید.

۷. برنامه‌ی آزمایش ۳ را طوری تغییر دهید که برنامه‌ی فرزند پیش از پدر خاتمه یابد. آنگاه پس از اجرای برنامه‌ی فرزند، در خط فرمان دستور ps –al را بزنید و با توجه به ستون وضعیت فرایندها (S)، شناسه‌ی فرزند را بیابید. چطور می‌توان ثابت نمود که این شناسه حتما متعلق به فرزند است؟

آزمایش نهم

ارتباط بین فرایندها در سیستم عامل لینوکس

۱۰-۱- مقدمه

فرایندها برای هماهنگ‌سازی فعالیت‌هایشان با یکدیگر و با هسته[1] در ارتباط‌اند. به این ارتباط که از طریق مکانیزم‌های متعددی انجام می‌گیرد، ارتباط بین فرایندی یا IPC[2] گویند. در تعریفی جامع تر می‌توان IPC را مجموعه‌ای از تکنیک‌های تبادل داده، میان نخ‌های مختلف یک یا چند فرایند نیز تعریف کرد. توجه به این نکته ضروری است که این نخ‌ها لزوما مربوط به یک سیستم واحد نیستند و می‌توانند روی چند کامپیوتر مختلف متصل به یک شبکه در حال اجرا باشند. روش مورد استفاده برای IPC براساس پهنای باند و تاخیر[3] ارتباط بین نخ‌ها و همچنین نوع داده‌ی مبادله شده متفاوت است.

۱۰-۲- هدف

دانشجو پس از انجام این آزمایش مفاهیم زیر را فراخواهد گرفت:

- انواع روش‌های ارتباط بین فرایندها در لینوکس
- مفاهیم Pipe، Signal
- مفاهیم تکلمیلی مربوط به همزمانی فرایندها در لینوکس و fork

[1] Kernel
[2] InterProcess Communication
[3] Latency

۱۰-۳- پیش‌آگاهی

برای فراهم سازی محیطی به منظور همکاری بین فرایندها دلایل متعددی را می‌توان نام برد:

- اشتراک اطلاعات
- تسریع در محاسبه
- پیمانه‌ای بودن[1]
- تسهیل در عملکرد
- ...

نخ‌های مربوط به یک فرایند واحد، به دلیل دارا بودن فضای حافظه مشترک، به راحتی قادر به اشتراک گذاری اطلاعات هستند. اما فرایندها به طورکلی دارای فضای حافظه مجزایی می‌باشند و نمی‌توانند به راحتی با یکدیگر ارتباط برقرار کنند، در اینجاست که IPC اهمیت خود را نشان می‌دهد.

لینوکس از تعدادی مکانیزم‌های IPC پشتیبانی می‌کند که لوله (Pipe) و سیگنال، نمونه‌های مهم‌تر آن‌ها هستند. سه روش دیگر IPC در لینوکس در زیر آمده‌اند که در ابتدا با عنوان System V در unix پیاده‌سازی شده بودند:

- Semaphore

استفاده از این ابزار، بیشتر به منظور قفل کردن ناحیه بحرانی[2] است. ناحیه بحرانی ناحیه‌ای است که اگر دو فرایند در آن به طور همزمان فعال باشند ممکن است به علت برش‌های زمانی[3] متنوع که پدید می‌آید، مشکلی یا استثناء[4] یا ناسازگاری داده به وقوع بپیوندد و در نتیجه‌ی آن خروجی غیر قابل پیش بینی تولید گردد. این مشکل ممکن است در هزاران بار اجرای یک برنامه پیش نیاید ولی احتمال وقوع آن در برنامه‌هایی که در آن‌ها مشکل ناحیه بحرانی در تخصیص منابع وجود دارد صفر نیست. Semaphore با تعریف یک متغیر عمومی بین آن فرایندها، مشکل را حل می‌کند، بدین‌گونه که هر فرایند، پیش از وارد شدن به ناحیه‌ی بحرانی‌اش متغیری را بررسی می‌کند و اگر مثلا مقدار آن متغییر صفر بود، یعنی می‌تواند به آن ناحیه وارد گردد (فرایند دیگری داخل ناحیه نیست) و قبل از ورود نیز مقدار متغیر را تغییر می‌دهد.

- Message Queue

[1] Modularity
[2] Critical region
[3] Time Slice
[4] Exception

برای اتصال یک صف از پیام‌ها، بین دو فرایند به کار می‌رود که به وسیله آن بلوک‌های کوچکی به نام Message بین آن دو، به طور غیر همزمان[1] مبادله می‌گردد.

- Shared memory

وقتی فرایندها می‌خواهند مقادیر زیادی داده را بین هم با یک روش کارامد به اشتراک بگذارند، از این مکانیزم استفاده می‌گردد.

۱۰-۴- سیگنال‌ها[2]

یک راه بسیار آسان، ساده و گاهی مفید برای یک فرایند که می‌خواهد با دیگری ارتباط برقرار کند و به او دستور بدهد یا از او درخواستی بکند، سیگنال است. در واقع یک فرایند می‌تواند با تولید یک سیگنال[3] و تحویل آن به فرایند دیگر، این اعمال را انجام دهد. با این‌کار گرداننده‌ی سیگنال[4] در فرایند مقصد، فعال شده و فرایند می‌تواند آنرا تفسیر کند. در اصل سیگنال رخدادی است که در سیستم های لینوکس و یونیکس در جواب به شرایط خاصی روی می‌دهد و هر فرایند دیگری که آن را دریافت کند بر اساس نوع سیگنال دریافتی کار خاصی انجام می‌دهد. در لینوکس تمامی سیگنال‌ها در سرآیند signal.h تعریف شده‌اند و تمام سیگنال‌ها با سه حرف SIG آغاز می‌گردند و معادل با تمامی آن‌ها یک عدد صحیح تعریف شده که کار با سیگنال را در ارسال به عنوان پارامتر به یک تابع، تسهیل می‌کند. چند مثال از سیگنال‌ها همراه با توضیح آنها در زیر آمده است:

جدول ۱۰-۱ چند نمونه سیگنال در سیستم‌عامل لینوکس

Signal Name	Description
SIGCHLD	Child process has stopped or exited
SIGCONT	Continue executing, if stopped
SIGSTOP	Stop executing. Can’t be caught or ignored
SIGTSTP	Stop signal for Terminal
SIGTTIN	Background process trying to read
SIGTTOU	Background process trying to write

برای مثال، اگر یک فرایند بخواهد فرایندی دیگر را به حالت stop ببرد، یک سیگنال SIGSTOP را به آن می‌فرستد. برای ادامه‌ی کار، سیگنال متوقف شده باید سیگنال SIGCONT را دریافت کند. سؤال

[1] Asynchronous
[2] Signals
[3] Raise a signal
[4] Signal Handler

اینجاست که فرایندی که سیگنال را دریافت کرده چگونه بفهمد که منظور از سیگنال دریافتی چیست؟ جواب این است که اکثر سیگنال‌ها از قبل برای مقاصد خاصی تعریف شده‌اند و عملکرد فرایندها در قبال دریافت هر یک از آنها مشخص است، یعنی فرایندها برای هر سیگنال، یک گرداننده‌ی سیگنال پیش فرض دارند.

برای مثال، سیگنال SIGINT مربوط به توقف اجرای غیرطبیعی برنامه توسط کاربر است و هنگامی‌که کاربر در هنگام اجرای برنامه، کلیدهای CTRL+C را فشار دهد این سیگنال فرستاده می‌شود. بدین ترتیب گرداننده‌ی پیش فرض این سیگنال، فرایند را مجبور به خروج می‌کند.

آیا می‌توان ماهیت رفتار گرداننده سیگنال SIGINT را طوری تغییر داد که به هنگام دریافت این سیگنال مثلا پیامی‌روی صفحه چاپ شود؟ جواب مثبت است! یعنی می‌توان رفتار فرایند را در مقابل دریافت سیگنالی خاص، به آنچه می‌خواهیم تغییر دهیم. اما این تمامی ماجرا نیست چون برای برخی سیگنال‌ها، نمی‌توان گرداننده‌ی آن را، غیر از گرداننده‌ی پیش فرض تعریف نمود. SIGKILL و SIGSTOP که به ترتیب برای کشتن و متوقف کردن فرایندها هستند، دو نمونه از این‌گونه سیگنال‌ها می‌باشند. دستور معادل سیگنال SIGKILL به صورت زیر است:

```
KILL -9 nnnn
```

که nnnn شناسه‌ی فرایند[1] و 9- بیانگر استفاده از سیگنال SIGKILL برای کشتن آن است. اما اگر بخواهیم بدون مشخص کردن سیگنال، فرایند را بکشیم، کافی است گزینه‌ی 9- را از دستور فوق حذف کنیم، در این صورت، حالت پیش‌فرض اتفاق می‌افتد. یعنی از سیگنال SIGTERM برای کشتن فرایند استفاده خواهد شد.

سوال بعدی که ممکن است در ذهن هر خواننده‌ای به وجود آید این است که آیا می‌توان سیگنالی فرستاد که تنها خودتان معنی آنرا می‌دانید؟ یعنی برای این سیگنال از قبل هیچ گرداننده‌ای تعریف نشده باشد؟ جواب این سؤال نیز مثبت است. دو سیگنال وجود دارند که رزرو نشده‌اند و عملکردشان از قبل تعریف نشده و شما در به خدمت گرفتن آن‌ها به هر قصدی که دارید کاملا آزادید. این سیگنال‌ها عبارتند از: SIGUSR1 و SIGUSR2. لیست نام سیگنال‌های مهم در زیر آمده است:

```
 1) SIGHUP    2) SIGINT    3) SIGQUIT   4) SIGILL
 5) SIGTRAP   6) SIGIOT    7) SIGBUS    8) SIGFPE
 9) SIGKILL  10) SIGUSR1  11) SIGSEGV  12) SIGUSR2
13) SIGPIPE  14) SIGALRM  15) SIGTERM  17) SIGCHLD
18) SIGCONT  19) SIGSTOP  20) SIGTSTP  21) SIGTTIN
22) SIGTTOU  23) SIGURG   24) SIGXCPU  25) SIGXFSZ
```

[1] Process ID

```
26) SIGVTALRM 27) SIGPROF  28) SIGWINCH 29) SIGIO
30) SIGPWR
```

۱۰-۵- تعریف و استفاده از یک گرداننده سیگنال

چگونه یک گرداننده‌ی سیگنال را برای یک سیگنال تعریف کنیم تا در هنگام فرستاده شدن آن، روندی که ما می‌خواهیم اجرا گردد؟ این کار را تابع ()signalکه در کتابخانه signal.h وجود دارد انجام می‌دهد. این تابع دو پارامتر می‌گیرد که اولی از نوع integer است و به آن، نام سیگنال را می‌فرستیم و دومی‌یک اشاره‌گر به تابع است که اسم تابع گرداننده سیگنالی که تعریف کرده‌ایم به آن فرستاده می‌شود. طرح اولیه‌ی[1] تعریف تابع signal، به صورت زیر است:

```
void *signal(int sig, void *func (int))
```

به طور مثال اگر بخواهیم به یک برنامه‌ی در حال اجرا، سیگنال SIGSTOP که باعث توقف آن با همان حالت می‌شود را بفرستیم، ولی برنامه در هنگام دریافت سیگنال فوق به جای توقف، تابعی را که ما نوشته ایم اجرا کند بدین صورت باید گرداننده‌ی دلخواه را برای این سیگنال تعریف کرد:

```
signal(SIGSTOP , user_defined_function);
```

که در این مثال عبارت user_defined_function نام تابع گرداننده‌ای است که می‌خواهیم وقتی فرایندی سیگنال SIGSTOP را می‌گیرد، اجرا شود. توجه شود که ذکر نام تابع با پرانتز اشتباه است:

```
signal(SIGSTOP,user_defined_function ());
// this is an ERROR !
```

توجه: سیگنال SIGSTOP را در لینوکس می‌توان با فشردن کلیدهای ctrl+z به برنامه ارسال کرد که در نتیجه آن اجرای فرایند متوقف می‌شود ولی فرایند از بین نمی‌رود و در حافظه اصلی می‌ماند.

۱۰-۶- توابع kill و raise

() kill یک فراخوانی سیستمی[2] است که دو آرگومان می‌گیرد: شماره سیگنال، که یک integer است و شماره فرایندی که سیگنال را به آن می‌فرستیم. با فراخوانی این تابع، سیگنال فراخوانده شده به آن فرایند فرستاده می‌شود. به طور مثال دستور زیر را در نظر بگیرید:

```
kill(3582,SIGSTOP);
```

این دستور سیگنال SIGSTOP را به فرایند، با شماره ۳۵۸۲ می‌فرستد. بدین ترتیب برخلاف تصور ظاهری از عملکرد فراخوان سیستمی kill می‌توان کارهای متنوعی غیر از کشتن فرایند، با آن انجام داد. همچنین

[1] Prototype
[2] System Call

تابع () raise کار مشابهی برای فرستادن یک سیگنال در محیط داخل یک فرایند انجام می‌دهد. مثلا بین چند نخ از یک فرایند واحد.

۱۰-۷- نکاتی از تابع () fork

اگر شما در فکر ساختن یک فرایند فرزند باشید به راحتی آنرا با تابع ()fork می‌سازید. اما فرایند ساخته شده به کار خود مشغول می‌شود و ارتباطی با بقیه ندارد. حال اگر بخواهید بین این فرایند و بقیه فرایندها ارتباط برقرار کنید باید از تکنیک‌های IPC بهره بگیرید.

وقتی فرایندی می‌میرد در واقع به طور کامل از بین نمی‌رود، آن فرایند مرده و دیگر اجرا نمی‌شود اما باقیمانده کوچکی از آن بر جای مانده که منتظر آن است که فرایند پدر آنرا بردارد. از جمله محتویات این باقیمانده، مقدار برگشتی فرزند است. به همین دلیل وقتی یک فرایند اقدام به اجرای ()fork می‌کند باید پس از تولد فرزندش منتظر اتمام آن شود تا بقایای آن‌را بردارد. اگر فرایند پدر قبل از فرزند بمیرد، فرآیند فرزند به فرزندخواندگی فرایند init[1] درمی‌آید اما اگر فرایند فرزند زودتر بمیرد تا هنگامی که فرایند پدر زنده است، در جدول فرایندها با عنوان <defunct> ثبت می‌شود که این بدین معناست که فرایند فرزند مرده، اما بقایای آن هنوز در سیستم باقی است و منتظر اتمام پدر است تا آن‌را با خود خاتمه دهد. در واقع استثنائی برای این حالت وجود دارد و آن این است که فرایند والد سیگنال SIGCLD را (که برای شناخته شدن فرزند به عنوان defunct تولید می‌شود) نادیده بگیرد که در نتیجه فرایند فرزند مجبور نیست منتظر اتمام پدر بماند. این روال استثنا به صورت زیر پیاده‌سازی می‌گردد:

```
#include <signal.h>
main()
{
          Signal (SIGCLD,SIG_IGN);
          //Now my child don't have to wait.
          .
          fork() ;
          .
          .
}
```

لازم به توجه است، در برنامه‌ی فوق SIG_IGN خود نوعی گرداننده‌ی سیگنال است که برای چشم‌پوشی[2] از سیگنال ذکر شده در ورودی همان تابع، مورد استفاده قرار می‌گیرد.

[1] فرایند با شماره PID برابر با ۱

[2] Ignore

در حالتی که فرایند والد قبل از اینکه برای فرایند فرزند به حالت wait برود بمیرد، فرایند فرزند به فرزندخواندگی فرایند init در می‌آید. در این وضعیت اگر فرایند فرزند هنوز زنده باشد، مشکلی پیش نمی‌آید اما اگر در حالت defunct باشد، بسته به عملکرد سیستم‌عامل لینوکس دو حالت ممکن است پیش آید:

- فرایند init به صورت پریودیک همه فرزندهایش را که در حالت defunct هستند از بین می‌برد.
- Init فرایند فرزند را به فرزند خواندگی قبول نمی‌کند.

در سیستم هایی که حالت ۲ را بروز می‌دهند، می‌توان با قرار دادن یک loop و گذاشتن فرایندهای defunct در جدول فرایندها و پر شدن ظرفیت این جدول، سیستم را مجبور به شروع به‌کار مجدد[1] کرد.

وضعیتی که فرایند فرزند در حالت خروج معمول دارد، یعنی خروج با استفاده از تابع () exit، را می‌توان با استفاده از تابع() wexitstatus بررسی کرد. وقتی فرزند روال ()exit را فراخوانی می‌کند، مقدار بازگشتی آن به والد برمیگردد که به صورت یک عدد integer است. اگر والد پیش از آن تمام شده باشد، به حالت wait رفته است تا روند اتمام فرزند نیز انجام گیرد. اگر این عدد را به تابع () wexitstatus به عنوان آرگومان بدهیم می‌توان اطلاعات حالتی که فرایند فرزند در حال خارج شدن داشته است را بدست آورد[2].

Pipes –۸–۱۰

تا کنون با سیگنال که یک راه ساده برای ارتباط بین دو فرایند است آشنا شدیم، اما در روش سیگنال، اطلاعاتی که از یک فرایند به دیگری می‌رسید محدود به یک عدد بود که شماره آن سیگنال را معرفی می‌کرد. در این قسمت با یک راه دیگر از مکانیزم‌های IPC آشنا می‌شویم که اجازه‌ی تبادل اطلاعات مفیدتر را به فرایندها می‌دهد. از لغت pipe یا لوله برای مفهوم زیر استفاده می‌شود:

< اتصال یک جریان داده‌اي از یک فرایند به فرایند دیگر>

در اصل خروجی یک فرایند را به ورودی دیگری متصل یا به اصطلاح پایپ می‌کنیم. اغلب کاربران لینوکس با کاراکتر پایپ[3] آشنا هستند که دو دستور shell را به هم متصل می‌کند به طوری که خروجی یکی به عنوان ورودی به دیگری تزریق می‌شود. به عنوان مثال دستور زیر را در نظر بگیرید :

[1] Reboot

[2] توجه داشته باشید که این تابع را باید بعد از اینکه تابع () wait تمام شد در ادامه دستورات فرایند والد اجرا کنید.

[3] Pipe character یا همان کاراکتر '|'

cmd1 | cmd2

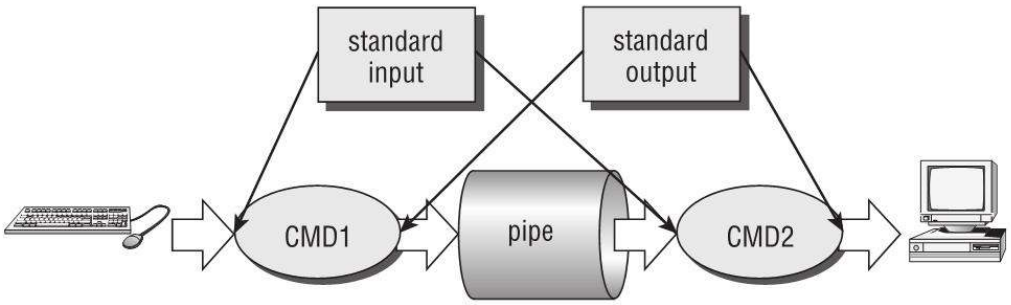
shell ورودی و خروجی دو دستور را بدین صورت می‌چیند:

۱. ورودی استاندارد cmd1 از صفحه‌کلید وارد می‌شود.

۲. خروجی استاندارد cmd1 به عنوان ورودی به cmd2 تزریق می‌گردد.

۳. خروجی استاندارد cmd2 به صفحه نمایش ارسال می‌گردد .

شکل متناظر با این مراحل در زیر مشاهده می‌شود:



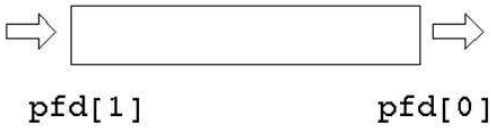
۱۰-۱مثالی از لوله در لینوکس

هیچ یک از فرم‌های IPC به اندازه PIPES ساده نیست. PIPES به همراه () fork یک روش خوب برای یادگیری مبانی IPC است. ابتدا لازم است آشنایی بیشتری با توصیف کننده های فایل[1] حاصل شود. توابع آشنای () fopen، () fclose و () fwrite که در کتابخانه stdio.h در زبان برنامه‌نویسی C هستند، برای کار با فایل‌ها مورد استفاده قرار می‌گیرند. این‌ها توابع سطح بالایی هستند که با استفاده از توصیف کننده‌های فایل، پیاده‌سازی می‌گردند و از فراخوانی‌های سیستمی همانند () open، () close و () write استفاده می‌کنند. این فراخوانی‌ها، معادل اعداد صحیحی هستند که با همان توابع سطح بالای نامبرده در stdio.h مشابه‌اند. برای مثال مقدار برگردانده شده در stdin، برای توصیف‌کننده فایل "۰"، stdout برابر "۱" و stderr برابر "۲" هستند. همچنین هر فایلی که به طور مثال توسط () fopen باز می‌شود، توصیف کننده‌های فایل خودش را می‌گیرد، گرچه این جزئیات از دید کاربر پنهان است.

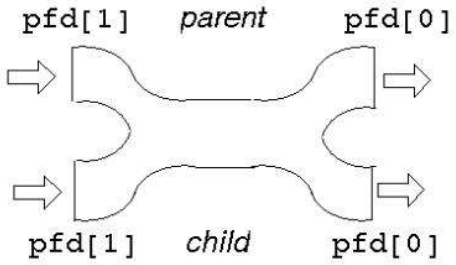
اساسا یک فراخوانی () pipe نیز یک جفت توصیف کننده فایل را بر می‌گرداند که یکی از آنها به رأس نوشتن[2] و دیگری به رأس خواندن[3] متصل‌اند. هر چیزی می‌تواند در pipe نوشته شود و از رأس دیگر آن به ترتیبی که به آن وارد شده خوانده شود. در اکثر سیستم‌ها pipe در صورتی که چیزی از آن نخوانیم، بعد

[1] File Descriptors
[2] Write end
[3] Read end

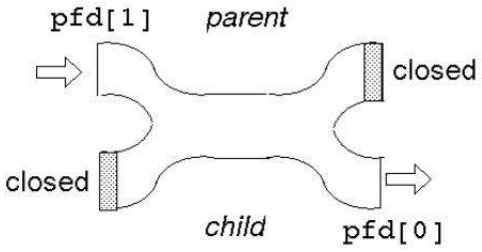
از نوشتن 10kb پر می‌شود. دو فرایند می‌توانند برای ارتباط با هم از تکنیک pipe استفاده کنند. فرض کنید که یک فرایند والد بخواهد برای فرزندش یک پیام بفرستد. برای این کار ابتدا و قبل از ایجاد فرزندش یک pipe ایجاد می‌کند؛ روندی مشابه به شکل زیر:



توجه کنید که pfd[1] را به توصیف کننده‌ی فایل مختص نوشتن و دیگری را به توصیف کننده فایل مختص خواندن اختصاص داده‌ایم. حال والد با فراخوانی fork () فرزندش را ایجاد می‌کند که در نتیجه‌ی آن، شکل بالا به این صورت تغییر می‌کند:



دلیل این است که پس از ایجاد فرزند، pipe نیز به اشتراک گذارده می‌شود. برای ارسال پیام از والد به فرزند، والد ابتدا ورودی read خود را بسته و ورودی write فرزند را باز می‌گذارد و سپس داده‌ها را وارد کانال می‌کند، آنگاه write خود را می‌بندد که با این کار به صورت خودکار، eof به فرزند فرستاده و فرزند نیز ورودی read خود را می‌بندد. هنگام ارسال و قبل از پایان بارگذاری داده‌ها در pipe، ساختار کانال بدین صورت است:



۱۰-۹- مراجع

1. Neil matthew and Richard stones,Beginning Linux ProgrammingT 4'th ed.,wiley Publishing, Inc., 2007.
2. Beej's Guide to Unix Interprocess Communication by Brian "Beej" Hall: http://www.ecst.csuchico.edu/~beej/guide/ipc/
3. Workshop on Inter Process Communication-IVEtsing YI-Department of Information Communication Technology
4. The Linux Programmer's Guide-BY : Sven Goldt,Sven van der Meer,Scott Burkett,Matt Welsh- Version 0.4 ., March 1995
5. CprE 308 Lab 5: Inter Process Communication in Unix Department of Electrical and Computer Engineering Iowa State University Spring 2006

۱۰-۱۰- دستور کار

۱. برنامه زیر را کامپایل و اجرا کنید.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   while (1)
   {
        getchar();
   }
   return 0;
}
```

الف) چند کاراکتر را در چند خط در محیط برنامه وارد کنید. حال کلیدهای ctrl+c را فشار دهید. چه اتفاقی می‌افتد؟ به نظر شما چه عاملی و چگونه باعث این اتفاق شد؟

ب) روش‌های ipc برای ارتباط بین دو فرایند به کار می‌روند، به نظر شما با فشار دادن کلیدهای ctrl+c چه قسمتی از سیستم عامل و از طریق چه روشی با برنامه قسمت الف ارتباط برقرار کرد؟

۲. برنامه بالا را اینگونه تغییر دهید و دوباره آنرا اجرا کنید.

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
void func(int);
int main ()
{
   if(signal(SIGINT,func)==SIG_ERR)
   {
        perror("Signal Error");
        exit(1);
   }

   while (1)
   {
        getchar();
   }
   return 0;
}
void func (int sig)
```

```
{
   printf("\nYou are not able to stop this yet !\n");
}
```

الف) حال با یک بار فشردن کلیدهای ctrl+c چه اتفاقی می‌افتد؟

ب) آیا اکنون می‌توان به اجرای برنامه ادامه داد؟ چرا؟

ج) حال یک بار دیگر برنامه را از نو اجرا کرده و این بار، دو مرتبه کلیدهای ctrl+c را فشار دهید. بار دوم چه اتفاقی می‌افتد؟ چه نتیجه‌ای می‌توان گرفت؟ چرا؟

۳. تابع گرداننده سیگنال را در آزمایش قبل بدین صورت تغییر دهید.

```
void func (int sig)
{
     signal (SIGINT,func);
     printf("\nyou are not able to stop this yet !\n");
}
```

الف) آیا اکنون با فشار دادن متوالی کلیدهای ctrl+c قادر به متوقف کردن برنامه هستید؟ چرا؟

ب) با فشردن کلیدهای ctrl+\ برنامه را متوقف کنید. با این کار شما سیگنال SIGQUIT را به برنامه ارسال کرده‌اید. برنامه را تغییر دهید به طوری که با فشردن کلیدهای فوق تغییری مشاهده نشود. می‌توانید برای این کار از سیگنال SIG_IGN به جای تعریف یک گرداننده جدید استفاده کنید. این گونه تعریف کردن گرداننده برای یک سیگنال، چه تفاوتی با تعریف کردن یک تابع به عنوان گرداننده دارد؟

ج) به نظر شما این کار (تغییر ماهیت اینگونه سیگنال‌ها) چه سود یا زیانی می‌تواند داشته باشد؟

د) تابع گرداننده‌ی سیگنال بالا را به گونه‌ای تغییر دهید که کاربر پس از چهار بار فشردن کلیدهای ctrl+c قادر به متوقف کردن برنامه شود. گرداننده سیگنال جدید را در برگه گزارش خود بنویسید.

(راهنمایی : از مفهوم متغیرهای استاتیک استفاده کنید)

۴. سیگنال SIGFPE در هنگام تقسیم یک عدد بر صفر به برنامه ارسال می‌گردد.

الف) برنامه‌ای بنویسید که در آن یک عدد بر صفر تقسیم گردد. حال توسط مفاهیمی که تا به حال آموخته اید کاری کنید که در هنگام رخ دادن تقسیم فوق عبارت "WARNING: This is a division by zero" بر صفحه ظاهر گردد و سپس از برنامه خارج گردد.

ب) به نظر شما کاربرد گزینه‌ی الف در چیست؟

۵. برنامه‌ای بنویسید که از شما یک کاراکتر بگیرد و پس از زدن enter اگر کاراکتر وارد شده حرف "q" بود با فرستادن سیگنالی توسط تابع kill، برنامه فورا بسته شود و در غیر این صورت عبارت " I like linux Operating System Laboratory" را چاپ کند.

راهنمایی: از تابع getpid() برای بدست آوردن شماره فرایند جاری استفاده کنید.

۶. برنامه‌ای بنویسید که در آن فرایند پدر زودتر از فرزند بمیرد. برنامه را در پس زمینه اجرا کنید و قبل از اینکه فرایند فرزند بمیرد دستور ps –al را اجرا کنید. پدر فرایند پدر چه کسی است؟ چگونه به این نتیجه رسیدید ؟

۷. برنامه‌ی آزمایش ۶ را طوری تغییر دهید که فرایند فرزند زودتر بمیرد. دو باره برنامه را اجرا کنید، در اجرای اول قبل از ایجاد فرزند، دستور signal (SIGCLD , SIG_IGN) ; را بنویسید و در اجرای دوم بدون این دستور برنامه را اجرا کنید. در هر اجرا برنامه را در پس زمینه اجرا کنید و قبل از اتمام فرایند پدر، دستور ps –al را بزنید .چه تفاوتی در خروجی این دستور در این دو بار اجرا مشاهده می-کنید ؟

۸. برنامه زیر را اجرا کنید:

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
int main()
    {
     int pfds[2];
     char buf[30];
     if (pipe(pfds) == -1) {
         perror("pipe");
         exit(1);
      }
     printf("writing to file descriptor #%d\n", pfds[1]);
     write(pfds[1], "test", 5);
     printf("reading from file descriptor #%d\n", pfds[0]);
     read(pfds[0], buf, 5);
     printf("read \"%s\"\n", buf);
     return 0;
}
```

الف) خروجی برنامه را بنویسید و به طور خلاصه برداشت خود را از خروجی شرح دهید.

ب) بعد از اجرای تابع pipe عناصر آرایه pfds مقداردهی می‌شوند. به نظر شما مقادیر داده شده به این آرایه چه اطلاعاتی به ما می‌دهند؟

ج) اعداد داده شده به توابع read و write چه کاری انجام میدهد ؟با تغییر ورودی های این توابع حدس بزنید که مقدار این اعداد چگونه باید باشد ؟

۹. برنامه زیر را کامپایل و اجرا کنید :

```
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main(){
     int pfds[2];
     char buf[30];
     pipe(pfds);

     if (!fork()) {
         printf(" CHILD: writing to the pipe\n");
         write(pfds[1], "test", 5);
         printf(" CHILD: exiting\n");
         exit(0);
      } else {
         printf("PARENT: reading from pipe\n");
         read(pfds[0], buf, 5);
         printf("PARENT: read \"%s\"\n", buf);
         wait(NULL);
      }
}
```

الف) در مورد خروجی برنامه به طور مختصر شرح دهید.

ب) با توجه به آموخته های جلسات پیشین در مورد شرط دستور if بالا توضیح دهید.

ج) اگر در برنامه بالا fork را قبل از pipe اجرا میکردیم آیا در روند برنامه مشکلی پیش می‌آمد ؟چرا ؟

آزمایش نهم

برنامه‌نویسی پوسته لینوکس

۱۱-۱- مقدمه

برنامه‌نویسی پوسته[1] در لینوکس شبیه به ویندوز است با این تفاوت که این امکان در ویندوزهای پیش از نسخه هفت، ضعیف و محدود است. این نوع برنامه‌نویسی انجام بسیاری از کارهای تکراری مانند تهیه نسخه پشتیبان[2] از پایگاه داده را که کاربر یا راهبر شبکه هر روز ممکن است انجام دهد تسهیل می‌کند. در مورد پوسته، نام بردن از مزیتی بدیهی مانند امکان استفاده از دستورات قبلی با کلید مکان‌نما به وضوح نشان می‌دهد که چقدر هدف بر تسهیل امور تکراری بوده است. این آزمایش به کلیات برنامه‌نویسی پوسته در لینوکس خواهدپرداخت.

۱۱-۲- هدف

آشنایی با برنامه نویسی پوسته لینوکس و ایجاد شناخت در رابطه با کلیت برنامه‌نویسی پوسته در این سیستم‌عامل.

---

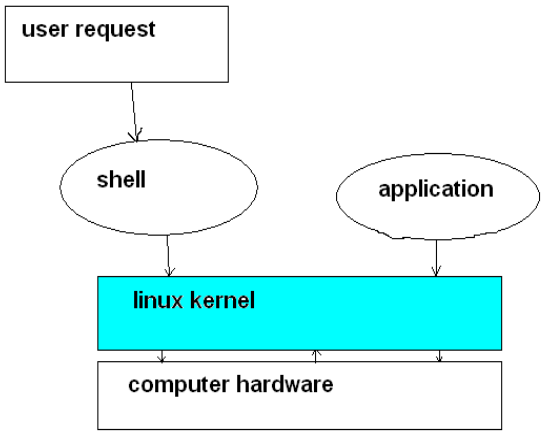
[1] Shell
[2] Back up

۱۱-۳- پیش‌آگاهی

پیش از پرداختن به مقدمات برنامه‌نویسی پوسته در لینوکس، اطلاع از ماهیت هسته و پوسته در این سیستم‌عامل و تفاوت‌های آن‌ها مفید خواهد بود.

۱۱-۳-۱- هسته چیست؟

Kernel یا هسته قلب سیستم عامل لینوکس است. هسته منابع سیستم عامل لینوکس را مدیریت می‌کند. منابع لفظ عامی است و به کلیه‌ی امکاناتی که سیستم‌عامل لینوکس در اختیار کاربر قرار می‌دهد اشاره دارد. به عنوان مثال دستگاه‌های ذخیره‌ی اطلاعات، چاپ آن، حافظه، مدیریت فایل و غیره را می‌توان نام برد. هسته تصمیم می‌گیرد چه کسی این منابع را چه مدت و تا چه زمانی استفاده کند و عملا اجرای برنامه‌های کاربر بر عهده‌ی هسته است. هسته واسطی است بین سخت افزار و برنامه‌ها (برنامه‌های کاربردی و پوسته). شکل زیر را مشاهده کنید.



شکل ۱۱-۱- نقش هسته در سیستم‌عامل لینوکس

هسته‌ی سیستم‌عامل لینوکس پس از بارگذاری شدن در حافظه، کارهای زیر را انجام می‌دهد:

۱- مدیریت ورودی خروجی

۲- مدیریت فرایندها

۳- مدیریت دستگاه‌ها

۴- مدیریت پرونده‌ها[1]

۵- مدیریت حافظه

[1] Files

۱۱-۳-۲- پوسته[1] چیست؟

زبان رایانه زبان ۰ و ۱ یا دودویی است. در اولین کامپیوترها دستورات در قالب دودویی به رایانه داده می‌شدند، که خواندن و نوشتن آن‌ها امری دشوار بود. به زبان ساده، برنامه‌ای به نام پوسته فراهم شد تا دستورات و فرامین به زبانی سطح‌بالاتر یعنی نزدیک به زبان طبیعی انگلیسی دریافت شود و در صورت معتبر بودن به هسته ارسال شود.

پوسته یک برنامه یا محیط است که برای تعامل با کاربر فراهم شده است. پوسته یک مفسر زبان دستورات است که فرامینی را که از یک وسیله ورودی مانند صفحه کلید یا از یک فایل دریافت کرده است، اجرا می‌کند. پوسته جزئی از هسته نیست اما از هسته استفاده می‌کند تا دستورات مستقیم کاربر را برای کارهای مختلفی مانند ایجاد فایل استفاده کند. در لینوکس پوسته وجود دارد، که در جدول زیر به صورت خلاصه به آن‌ها اشاره شده است:

جدول ۱۱-۱- انواع پوسته در لینوکس

Shell Name	Developed by	Where	Remark
BASH(Bourne-Again Shell)	Brian Fox and Chet Ramey	Free Software Foundation	Most common shell in Linux .It's Freeware Shell.
CSH(C shell)	Bill Joy	University of California(For BSD)	The C shell's syntax and usage are very similar to the C programming language
KSH(Korn Shell)	David Korn	AT&T Bell Labs	--
TCSH	See the man page Type $ man tcsh	--	TCSH is an enhanced but completely compatible version of the Berkeley UNIX C shell(CSH)

از نکات مهم مرتبط با این جدول لازم است اشاره شود که همه‌ی پوسته‌ها عملکرد یکسانی دارند اما هر یک نحو[2] خاص خود را دارند. برای اینکه بدانید لینوکس مورد استفاده‌ی شما چه پوسته‌هایی را پشتیبانی می‌کند دستور زیر را در خط فرمان تایپ کنید:

$ cat /etc/shells

لازم به یادآوری است که در سیستم‌عامل DOS که اکنون در سیستم‌عامل ویندوز نیز نسخه‌های مجازی آن وجود دارد(چرا مجازی؟)، پوسته توسط برنامه‌ای با نام Command.com است که عملکردی مشابه به پوسته‌ی لینوکس دارد اما از آن ضعیف‌تر است و امکانات کمتری دارد.

[1] Shell
[2] Syntax

۱۱-۳-۳- Shell Scripting در لینوکس

در حالت عادی پوسته دستورات را یکی یکی گرفته و اجرا می‌کند. برای مثال در برنامه‌ایی که ۲۰ دستور دارد، ابتدا باید دستور اول را نوشته و کلید اینتر را روی صفحه کلید بزنیم و پس از اجرای آن دستور دوم و الی آخر. اما اگر شما دستورات را تماما در یک فایل ذخیره کنید و سپس به پوسته بگویید که ازاین فایل به عنوان ورودی برای دستورات استفاده کند، این رویه Shell Scripting نام دارد. به عبارت بهتر Shell Scripting یک سری دستور هستند که در یک فایل متنی قرار دارند، همانند فایل‌های Batch در سیستم عامل ویندوز، با این تفاوت که قوی‌ترند.

از فواید Shell Scripting در لینوکس می‌توان به همان مزایا در ویندوز اشاره کرد، اما به صورت خلاصه گرفتن ورودی از کاربر و نمایش نتیجه بر خروجی یعنی پردازش بلادرنگ دستورات کاربر به سیستم‌عامل، صرفه جویی در زمان با ایجاد فایل‌های دسته‌ای و خودکارسازی بسیاری از فعالیت‌ها در رابطه با سیستم‌عامل به خصوص در محیط کار، از مزایای عمده‌ی Shell Scripting در لینوکس به حساب می‌آید.

۱۱-٤- نوشتن برنامه‌ها

برای شروع برنامه‌نویسی در لینوکس بایداز مسیر زیر به خط فرمان آن وارد شویم و در آنجا دستورات را بنویسیم:

Applications/Accessories/Terminal

همانطور که پیداست به محض تایپ هر دستور نتیجه آن را می بینیم این کار هدف اسکریپت نویسی نیست و مطلوب است برنامه را یک‌بار نوشته و آن را اجرا کنیم. لازمه این کار ورود به محیط VI است که کار با آن در آزمایش‌های قبلی توضیح داده شده است . اما راه حل ساده‌تر این است که وارد یک ویرایشگر متن شده و کد خود را نوشته و بعد با فرمت sh. آن را ذخیره کنید. توجه داشته باشید که استفاده از پسوند sh. اختیاری است و بیشتر به این خاطر است که فرد از ماهیت فایل اطلاع داشته باشد و گرنه می‌توان فایل Shell Script را بدون پسوند نیز نوشت و اجرا نمود.

۱۱-٤-۱- اولین برنامه

برنامه زیر یک فایل به نام first ایجاد کرده و دستور نمایش یک رشته روی صفحه نمایش را در آن صادر می‌کند.

```
$ vi first
#
# First shell script
#
clear
echo “Hello Shell Scripting”
```

در شرح برنامه لازم است ذکر شود که هر کلمه‌ای بعد از کاراکتر '#' بیاید، هنگام کامپایل نادیده گرفته می شود. به عبارت دیگر این کاراکتر نقش ایجاد Comment در برنامه را دارد. دلیل استفاده از توضیحات افزایش وضوح برنامه است. خط اول فایلی به نام first ایجاد می‌کند و دستور clear صفحه نمایش را پاک می‌کند. دستور آخر یا echo نیز رشته را در صفحه نمایش چاپ می‌کند.

حال از محیط vi خارج شوید و دستور first/. را تایپ کنید تا برنامه اجرا شود. درصورتی که خطای Permission denied نمایش داده شد، یعنی اجازه دسترسی به این فایل به شما داده نشده است با تغییر نحوه دسترسی به این فایل می‌توانید آن را اجرا کنید .دستور زیر این کار را برای شما انجام می‌دهد:

```
chmod +x first
```

پس از آن اگر first/. را تایپ کنید برنامه شما اجرا می‌شود .

نکته:

- هر زمان خواستیم به خط جاری پایان دهیم و ادامه‌ی آن را در خط بعد بنویسیم از کاراکتر \ استفاده می‌کنیم:

```
echo \ "Salam"
```

- هر جا که به خط بعد رفتیم معادل کاراکتر ; حساب می‌شود و بلعکس. یعنی دوبخش زیر معادلند:

if true then echo "Condition is true" fi	if true; then echo "Condition is true"; fi

۱۱-٤-۲- متغیرها در پوسته

در سیستم عامل لینوکس ۲ نوع متغیروجود دارد: نوع اول متغیرهای سیستمی که به وسیله‌ی خود سیستم عامل ایجاد و مدیریت می‌شوند. این متغیرها با حروف بزرگ تعریف می‌شوند و نوع دوم که متغیرهای

کاربر یا UDV[1] که به‌وسیله‌ی کاربر ایجاد و مدیریت می‌شوند. این متغیرها با حروف کوچک تعریف می‌شوند. برای تعریف متغیرهای کاربر از نحو زیر پیروی می‌کنیم:

$variable name=value

value مقداری است که به متغیری به اسم variable name داده می‌شود. باید در نظر داشت که نام متغییر در Shell Scripting در سیستم‌عامل لینوکس حتما با علامت دلار $ شروع می‌شود. اما سایر قواعد متعارف نامگذاری متغییرها در سایر زبان‌های برنامه‌نویسی در اینجا نیز صدق می‌کند. برای مثال دستور $no=10 متغییری به نام no با مقدار ۱۰ ایجاد خواهد کرد.

قوانین نامگذاری متغیرها در سیستم‌عامل لینوکس مشتمل بر موارد زیر است:

۱. اسامی متغیرها با یک حرف از الفبای انگلیسی یا کاراکتر '_' شروع می‌شود و با یک یا بیشتر کاراکتر ادامه می‌یابد.

۲. فضای خالی (Space) در هیچ یک از طرفین مساوی قرار داده نشود. به عنوان مثال عبارات زیر نادرستند:

```
$ no  =10
$no=  10
$10 = no
```

و فرم درست آن به صورت زیر است:

```
$no=10
```

۳. متغیرها به حروف کوچک و بزرگ حساس هستند.

۴. می‌توان متغیرهایی با مقدار تهی (null) تعریف کرد.

```
$ ver=""
```

۵. نمی‌توان از علامت‌های * و ? در نام متغیرها استفاده کرد.

۶. تعریف عدد صحیح در لینوکس به صورت زیر است.

```
declare –i a=5
```

دستور echo

وظیفه‌ی اصلی این دستور همان نمایش پیغام‌ها و متغیرهاست و نحو آن به شکل زیر است:

```
echo [option] [string,variable]
```

[1] User Defined Variable

با این دستور کارهای زیادی می‌توان انجام داد. برای مثال رنگی کردن متن، پررنگ کردن، خاکستری کردن، رفتن به خط بعدی و .... .برای مثال دستور زیر سبب می‌شود نوشته به رنگ آبی نشان داده شود.

```
$echo –e "\033[34m Hello OSLab!"
```

عبارت echo در دستور بالا از الگوی ANSI استفاده می‌کند. یعنی با نوشتن 033\ در ابتدا، سبب انجام عملی می‌شویم که 34m] به آن عمل اشاره دارد و بیانگر استفاده از رنگ آبی برای حروفی است که قرار است نمایش داده شوند و آن حروف نیز !Hello OSLab هستند. برای ایجاد سایر اثرها مانند پررنگ کردن، چشمک زدن حروف و ... بایستی با الگوی ANSI مشابه به فوق یعنی]033\ شروع کرد و بعد از آن یک عدد و سپس یکی از حروف m,q,s,u را استفاده نمود که تنها برخی از این اعداد معتبر می‌باشند.

جدول ۱۱-۲- گزینه‌های دستور echo

Character or Letter	Use in CSI	Example
h	Set the ANSI mode	**Echo –e"\033[h"**
l	Clear the ANSI mode	**Echo –e "\033[l"**
m	Useful to show characters in different colors or effects such as BOLD and Blink	**Echo –e "\033[35m Hello"**
q	Turns keyboard num lock,caps lock,scroll lock LED on or off	**Echo –e"\033[2q"**
s	Store the current cursor x,y position(col,row position) and attributes	**Echo –e "\033[7s"**
u	Restore cursor position and attributes	**Echo –e "\033[8u"**

۱۱-٤-۳- عبارات ریاضی در پوسته

برای استفاده از عبارات ریاضی در لینوکس باید از نحو زیر پیروی کنیم:

```
expr op1 math-operator op2
```

مثال:

```
$ expr  1 + 2
$ expr  10 \* 3
$echo `expr  6 + 3`
```

در اینجا اشاره به برخی نکات مهم ضروری است. مثلا در لینوکس علامت ضرب با *\ نشان داده می‌شود. همچنین در صورت عدم رعایت فاصله بین عملگر و متغییر یا عدد، ممکن است کل عبارت به شکل یک

رشته فرض شود. همچنین در دستور سوم از کاراکتر ` استفاده شده است که با کاراکتر ' متفاوت است. به عبارت دیگر، مکان کاراکتر `در بالای کلید Tab است!

هنگامی که دستوری به لینوکس داده می‌شود، لینوکس پس از انجام آن یک مقدار موسوم به وضعیت خروج[1] برمی‌گرداند. مقدار ۰ به معنای انجام موفقیت‌آمیز دستور و مقدار غیرصفر به معنای شکست در انجام آن است. چگونگی دستیابی به این متغیر ساده است. دستور زیر این کار را انجام می‌دهد:

$echo $?

دستور read

این دستور برای دریافت ورودی از کاربر است. نحو آن به شکل زیر می‌باشد:

read variable1, variable2,…, variable

برنامه زیر از کاربر نامش را درخواست کرده و سپس به او پیامی را مشتمل بر اسم کاربر نمایش نشان می‌دهد:

```
$ vi getname
# get user name
echo “Please  Enter Your  Name:”
read  name
echo “hello $name”
```

قالب دستورif

ساختار if به صورت زیر است:

```
if  condition
then  commands
elif   condition
then  commands
else  commands
fi
```

به جای condition شرط را و به جای commands دستورات را وارد می‌کنیم. قسمت‌های elif و else اختیاری هستند.

[1] Exit Status

دستور test

این دستور برای مشاهده‌ی درستی یا نادرستی یک عبارت استفاده می‌شود. برای مثال مقایسه دو عدد و گرفتن نتیجه. نحو آن به صورت زیر است:

test expression

برای مثال:

```
if test 5 -gt 0
then
echo "first number is greater"
fi
```

شکل ساده‌تر استفاده از دستور test، جایگزینی آن با عبارت [ ] است. مانند برنامه‌زیر که معادل با برنامه‌ی فوق است:

```
if [ 5 -gt 0 ]
then
echo "first number is greater"
fi
```

دستور test نه تنها در مورد اعداد بلکه در مورد رشته‌ها و فایل‌ها نیز کاربرد دارد. در زیر برخی از کاربردهای(پارامترهای) آن را می‌بینید:

جدول ۱۱-۳- کاربردهای دستور test

عبارت	توضیحات
!exp	**اگر exp غلط باشد(not)**
exp1 -a exp2	**اگر exp1 و exp2 هردو درست باشند(and)**
exp1 -o exp2	**اگر exp1 یا exp2 یا هردو درست باشند (or)**
str1 = str2	**اگر str1 برابر str2 باشد**
str1 != str2	**اگر str1 مخالف str2 باشد**
int1 -eq int2	**اگر int1 برابر int2 باشد**
int1 -nq int2	**اگر int1 مخالف int2 باشد**
int1 -gt int2	**اگر int1 بزرگ‌تر از int2 باشد**
int1 -ge int2	**اگر int1 بزرگ‌تر یا مساوی int2 باشد**
int1-lt int2	**اگر int1 کوچک‌تر از int2 باشد**
int1 -le int2	**اگر int1 کوچک‌تر یا مساوی int2 باشد**
-e file	**اگر file وجود داشته باشد**

برعکس زبان‌هایی مانند خانواده‌ی C که خروجی صفر معادل با False (غلط) و غیر صفر معادل با True (درست) است، در Shell Scripting، صفر برابر True و غیر صفر برابر False است.

۱۱-۵- حلقه‌ها در اسکریپت‌نویسی پوسته

پوسته‌ی لینوکس از دو نوع حلقه زیر حمایت می‌کند:

for loop: قالب این حلقه به شکل زیر است:

```
for ((expr1;expr2;expr3))
do
……
done
```

While loop : در لینوکس حلقه while صورت زیر است:

```
while [ condition ]
do
    command1
    command2
    ….
done
```

در رابطه با دستورهای break وcontinue در مورد حلقه‌ها لازم به یادآوری است که با دستور continue دستورهای بعدی حلقه اجرا نمی‌شود و دوباره به اول حلقه منتقل می‌شویم، اما با دستور break، دستورهای بعدی حلقه اجرا نمی‌شود و برنامه از حلقه خارج می‌شود. اگر حلقه‌های تو در تو داشته باشیم و بخواهیم دو یا چند بار break یا continue را اجرا کنیم، در جلوی این دستورات تعداد تکرار را وارد می‌کنیم.

البته حلقه‌ها لزوما بر اعداد متکی نیستند و برنامه‌ی زیر نمونه مثالی است در رابطه با استفاده‌های دیگر حلقه‌ی for که * به تمام نام فایل‌های موجود در فهرست جاری اشاره دارد. بنابراین تمام اسم فایل‌ها در حلقه بررسی و چاپ می‌شوند.

```
for  fn  in *; do
        echo "$fn"
done
```

مثالی از کاربرد دستور while نیز که سبب چاپ اعداد ۱ تا ۶ در شش سطر مجزا می‌شود در زیر آمده است:

```
n=1
while [ $n –le 6 ]; do
                echo $n
```

```
                let n++
done
```

همانطور که می‌توان حدس زد، دستور let برای انجام عمل محاسباتی مورد استفاده قرار مـی‌گیـرد. نکتـه‌ی دیگر اینکه می‌توان به جای عبارت [ ] در شرط حلقه، از دستور test نیز استفاده کرد.

۱۱-٦- مراجع

1. Quigley, Ellie, **Linux Shells by example**, Prentice-Hall Inc., 2000.
2. Vivek G. Gite, **Linux shell scripting tutorial ver.1.0**, http://edoc.hu-berlin.de/oa/articles/reiP8SRiytnBs/PDF/21bUc8YeDzZpE.pdf

۱۱-۷- دستور کار

۱. متغییری به نام myno با مقدار اولیه‌ی ۱۰ تعریف کنید. سپس دستور زیر را بنویسید:

$echo myno

الف) خروجی دستور چیست؟

ب) برای اینکه مقدار ۱۰ به عنوان خروجی چاپ شود باید قالب دستور را چگونه تغییر دهیم. بنویسید.

۲. مقدار متغیر وضعیت خروج را برای هر یک از عبارات زیر بنویسید (برای هر مورد دستوراتی متناسب با آن نوشته و تست کنید):

الف) تقسیم بر صفر

ب) حذف فایلی که وجود ندارد

ج) انجام یک عبارت محاسباتی بدون خطا

۳. اشتباهات برنامه ی زیر را رفع کنید و آن را اجرا نمایید. صورت صحیح آن را در گزارش‌کار بنویسید.

```
$m  =8
$a?=2
if test $m gt a?
then
echo “first number is greater”
```

۴. با آزمون و خطا، کاربرد هر یک از دستورات زیر را هرکدام حداکثر در یک سطر بنویسید:

الف) whoami

ب) date

ج) cd ~

راهنمایی: برای درک عملکرد افزودن ~ به دستور cd بهتر است ابتدا سلسله دستورات زیر را انجام دهید و خروجی دو دستور pwd را مقایسه کنید.

```
$ cd /
$ pwd
$ cd ~
$ pwd
```

د) echo {a..z}

۵. الف) در لینوکس برای کار با هر آرگومان، عبارت $n به‌کار می‌رود که n از صفر شروع می‌شود. کار با آرگومان در لینوکس دقیقا مانند کار با آن در ویندوز است. تنها تفاوت آن‌ها در پیشوند است که در لینوکس علامت آن دلار است. برنامه‌ایی بنویسید که ۲ عدد را به عنوان آرگومان بگیرد و عدد بزرگتر را نمایش دهد.

ب)اگر به جای عدد، رشته را به عنوان ورودی برنامه وارد کنید چه پیغامی نمایش داده می‌شود.آن را در گزارش کار خود بنویسید.

۶. دستورات زیر را در یک فایل ذخیره نمایید و سپس اجرا کنید :

```
$ x=`/sbin/ifconfig`
$ echo $x
```

چه اطلاعاتی را از خروجی دریافت می‌کنید. چند مورد آشنا را ذکر کنید.

۷. برنامه ی زیررا اجرا کنید و نحوی عملکرد آن را شرح دهید.

```
s=”Linux OS”
for ((i=8 ;i>0 ;i--))
do
s=${s:0:i}
echo $s>>1.txt
done
```

۸. در لینوکس تابع به صورت زیر تعریف می‌شود:

```
function_name()
{
 command1;
 command2;
 ..
 commandN;
}
```

و نحوه‌ی فراخوانی تابع به صورت زیر است:

```
function_name
```

الف) با فرض اینکه برای کسب شناسه‌ی کاربر جاری از دستور id -u استفاده شود و شناسه‌ی کاربر ریشه ۰ باشد، تابعی بنویسید که id کاربر جاری را بدست آورده و با پیامی مشابه به user is root بگوید که کاربر root است یا نه؟

ب) اگر از محیط Sun Virtual Box استفاده می‌کنید، ابتدا سیستم‌عامل فعلی را Shut Down کرده و سپس بر آن کلیک راست کرده و ...settings را انتخاب کنید، آنگاه به مسیر زیر بروید:

Storage->IDE Controller ->Host Drive

در سمت راست و در قسمت Attributes درایو CD–ROM خود را به آن معرفی کنید و دوباره لینوکس را start کنید. تابع زیر را در فایل cd.sh ذخیره کرده و نتیجه‌ی اجر را در کاربرگ یادداشت کنید.

```
function lscd()
{

DEVICE=$1
mount $DEVICE
ls -l $DEVICE
umount $DEVICE
eject $DEVICE
}
```

۹. با استفاده از دستور while، برنامه‌ای بنویسید که ماتریس زیر را چاپ کند:

```
 1   2   3   4   5   6   7   8   9  10  11
 2   4   6   8  10  12  14  16  18  20  22
 3   6   9  12  15  18  21  24  27  30  33
 4   8  12  16  20  24  28  32  36  40  44
 5  10  15  20  25  30  35  40  45  50  55
 6  12  18  24  30  36  42  48  54  60  66
 7  14  21  28  35  42  49  56  63  70  77
 8  16  24  32  40  48  56  64  72  80  88
 9  18  27  36  45  54  63  72  81  90  99
10  20  30  40  50  60  70  80  90 100 110
11  22  33  44  55  66  77  88  99 110 121
```

راهنمایی: برای کنترل بیشتر بر چیدمان خروجی، می‌توان از دستور printf در زبان C همانند زیر استفاده کرد:

```
printf "\n" $y
```

آزمایش یازدهم

آشنایی و به‌کارگیری Samba Server 3.0 در لینوکس

۱-۱- مقدمه

Samba این امکان را ایجاد می‌کند تا کامپیوتر لینوکس مانند هر کامپیوتر دیگری در شبکه ویندوز، فایل‌ها و چاپگرهای خود را به اشتراک بگذارد. به علاوه، نسخه های جدید Samba، اغلب سرویس‌هایی که یک کارگزار ویندوز در یک شبکه ارائه می‌کند را در اختیار قرار می‌دهند. Samba برای مقاصد زیر به کار می‌رود:

- کارگزار[1] فایل و پرینتر برای سرویس گیرنده‌های[2] ویندوز و Samba
- کنترلگر دامنه[3] در یک شبکه ویندوز(تصدیق کاربر و ....)
- کارهای پیچیده دیگر، از قبیل استفاده از یک کنترلگر دامنه ویندوز برای معتبرسازی[4] کاربران لینوکسی

این آزمایش بر اساس Samab Server نسخه‌ی 3.0 تنظیم شده است.

---

[1] Server
[2] Clients
[3] Domain controller
[4] Validation

۱-۲- هدف

به طور کلی هدف از این آزمایش، آشنایی با سرویس رایگان Samba و نحوه نصب این سرویس و استفاده از آن جهت راه‌اندازی یک کارگزار فایل در لینوکس می‌باشد. در انتهای این آزمایش دانشجویان قادر به انجام فعالیت های زیر خواهند بود:

- توانایی انجام تنظیمات مربوط به شبکه لینوکس
- توانایی نصب Samba Server و انجام تنظیمات مربوط به آن
- توانایی به اشتراک‌گذاری فایل در شبکه مبتنی بر ویندوز با استفاده از File Server ایجاد شده در لینوکس به وسیله فعال‌سازی سرویس Samba
- آشنایی با ابزار Smbclient
- آشنایی با تنظیم از راه‌دور کارگزار Samba به وسیله ابزار SWAT

۱-۳- پیش‌آگاهی

Samba مجموعه ای از ابزارها و دیمون‌هایی[1] است که از پروتکل Smb[2] برای انجام کارهای مربوط به شبکه استفاده می‌نماید.

۱-۳-۱- تاریخچه

پروژه Samba در سال ۱۹۹۲ توسط AndrewTridgell شروع شد. Samba امکان به اشتراک گذاری فایل ها و پرینترها را با کامپیوترهای که مایکروسافت ویندوز را اجرا کنند،امکان پذیر می‌سازد. Samba از پروتکل شبکه SMB استفاده می کند.

۱-۳-۲- معرفی کارگزار Samba

تا این جا متوجه شده‌اید که Samba وظیفه ارتباط لینوکس و ویندوز را،به منظور انتقال فایل و به اشتراک بر عهده دارد ، اما چگونه؟به عبارت دیگر باید گفت که Samba روی اساس[3] یونیکس اجرا می گردد اما با یک Windows clinetبه زبان محلی خودش صحبت می کند و این توانایی به یک سیستم

[1] Deamons
[2] Server massage block
[3] Platform

یونیکس اجازه می دهد بدون این که صدمه‌ای به فایل ها و دایرکتوری‌ها بزنـد، وارد یـک شـبکه ویـنـدوز گردد. به این ترتیب کاربران شبکه ویندوز می‌توانند به راحتی به فایـل هـا و پرینتـر دسترسـی داشـته باشـند بدون این که حتی بدانند که این امکانات توسط بـک سـرور یـونیکس فـراهم گردیـده‌اسـت. همـه‌ی ایـن امکانات به خاطر وجود پروتوکلی به نام CIFS[1] می باشد و قلب این پروتکل ، پروتکل(SMB) می‌باشـد که مدت زیادی از عرضه آن می گذرد.

Samba یک نسخه پیاده سازی شده از CIFS می‌باشد. Samba تنها مختص به ویندوز نمی باشد.به طور مثال سیستم عامل OS/2 نیز از به اشتراک گذاری پرینتر و فایل تحت SMB پشتیبانی می کند و همچنین یک نسخه تجاری از آن به منظور استفاده در مکینتاش عرضه شده‌است.

در این مرحله سعی می‌شود کمی بیشتر و دقیق تر در مورد نحوه عملکرد Samba صحبت نماییم. Samba از دوزیر برنامه اساسی تشکیل شده‌است ، Smbd و Nmbd . که این دو در واقع دو پیاده‌سازی از چهار سرویس مهم، از نسخه به روز CIFS می‌باشند که شامل موارد زیر می گردد:

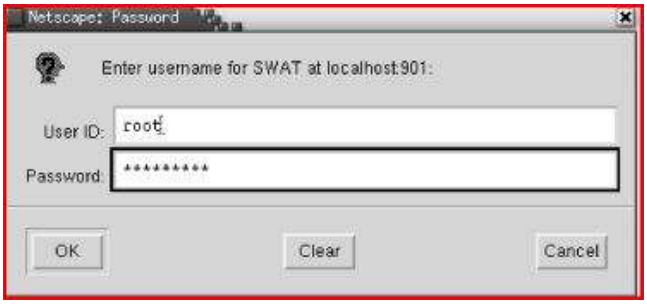
- File and printer services
- Authentication and authorization
- Name resolution
- Service amnouncement (browsing)

سرویس اول یعنی File and printer services که البته اصلی ترین سرویس CIFS می‌باشد توسط Smbd و یا همان SMB Daemon تامین می گردد.همچنین این مورد Share mode و User modeرا که همان Authentication and authorization را فراهم می کند.که در آن ساده ترین حالت به اشتراک گذاری ،اختصاص یک رمزعبور به آن فایل یا پرینتر می‌باشد که تنها افراد با داشتن آن رمز قادر به استفاده از آن‌ها می باشند.

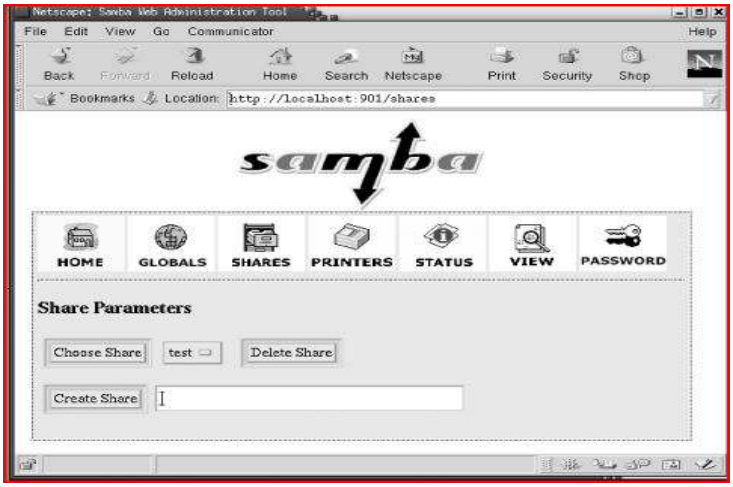
کاربردهای دیگر Samba عبارتند از :

- Smbclinet : این مورد ، کاربردی همانند FTP دارد.
- Nmblookup : یک کلاینت NetBIOS name service از این امکان می توان جهت یافتن NetBIOS در یک شبکه با استفاده از فریاد زدن IP استفاده نمود.
- Swat : با استفاده از امکان Samba Web Administrator Tool مي توان با استفاده از مرورگر وب، کارگزار samba را تنظیم نمود.

[1] Common Internet File System



شکل ۱۲-1 نمای صفحه ورود به SWAT



شکل ۱۲-۲ نمای محیط کار swat

۱-۳-۳- نحوه به اشتراک گذاری یک پوشه[1] به وسیله Samba

مهمترین فایلی که جهت پیکربندی Samba در این قسمت وجود دارد smb.conf است. این فایل شامل مهمترین تنظیمات کارگزار Samba برای به اشتراک گذاری فایل ها و پرینتر می‌باشد. حال برای تنظیم سرور دو راه وجود دارد :

۱. تنظیم گرافیکی
۲. تنظیم دستی

در تنظیم گرافیکی از واسط کاربری خود لینوکس جهت به اشتراک گذاری یک فایل استفاده می‌شود. اما در صورتی که بخواهیم کارگزار Samba را به صورت دستی تنظیم نماییم باید فایل /etc/samba/smb.conf را مطابق با نیاز خود ویرایش نماییم. در این فایل توضیحات[2] با #(شارپ) و

[1] Directory
[2] Comments

;(نقطه ویرگول) آغاز می‌شوند. دراین فایل هر عملیات به اشتراک گذاری با [ ] [1] علامت گذاری می‌شود و در ادامه آن با گزینه‌هایی که اکثرا به صورت دودویی[2]، با مقادیر Yes و No ، می‌باشند تنظیم می‌گردد.
بعد از باز کردن smb.conf مراحل زیر را جهت به اشتراک گذاری یک پوشه مشخص، باید دنبال کنید:
ابتدا Key/Value های زیر را در بخش Global به صورت زیر تغییر دهید :

نام شبکه خود را وارد نمایید = Workgroup

.
.
.
.
Security = user

بخش جدیدی در انتهای فایل به صورت زیر اضافه نمایید:

```
[share]
comment = Ubuntu File Server Share
path = <your own share directory path>
browseable = yes
guest ok = yes
read only = no
create mask = 0755
```

منظور از هرکدام از دستورات بالا در زیر آورده شده است:

- Comment: توصیفی از اشتراک جاری می‌باشد.
- Path : مسیر پوشه‌ای که قرار است به اشتراک گذاشته شود.
- Browseable : این گزینه امکان مشاهده پوشه‌های به اشتراک گذاشته شده را با استفاده از Windows Explorer به کاربران ویندوز می‌دهد.
- Guest ok : این گزینه به client اجازه می دهد تا بدون داشتن رمز عبور به پوشه به اشتراک گذاشته شده دسترسی داشته باشد.
- Read only : دادن اجازه نوشتن در پوشه به افرادی که دسترسی دارند.
- Create mask : سطح دسترسی های[3] پوشه را مشخص می‌نماید.

[1] براکت

[2] Boolean
[3] Permissions

حال باید پوشه مورد نظر را ایجاد و دسترسی‌های آن را تنظیم کنیم برای این کار وارد محیط ترمینال شده و دستورات زیر را وارد می‌کنیم:

```
sudo mkdir -p <your own share directory path>

sudo chown nobody.nogroup <your own share directory path>
```

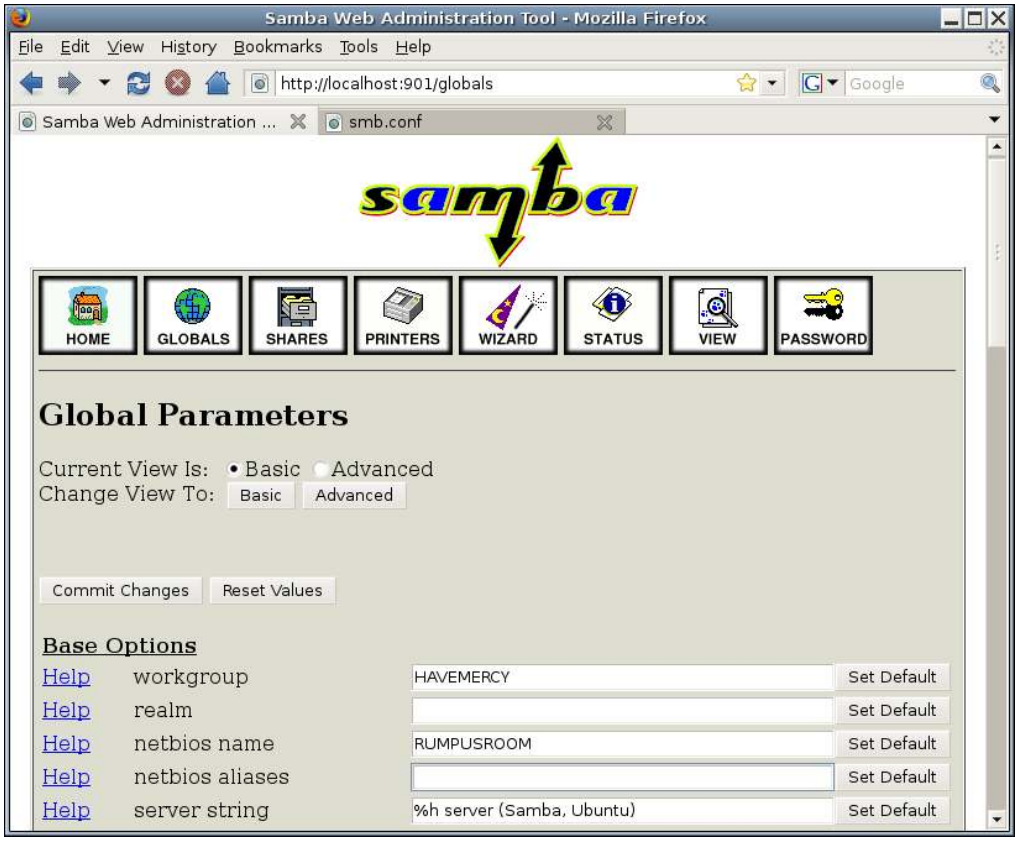
و در آخر سرویس Samba را باید راهندازی مجدد نماییم:

```
sudo /etc/init.d/samba restart
```

حال پوشه مورد نظر به اشتراک گذاشته شده و در شبکه ویندوزی ما قابل استفاده می‌باشد.

۱-۳-٤- آشنایی با ابزار SWAT

همان طورکه می‌دانید اکثر دستورات درLinux به صورت Command Shell اجرا می‌شوند. برای مدیران شبکه‌ای که با سیستم عامل Windows کار کرده‌اند راحت‌تر این است که تمامی تنظیمات کارگزار خود را از طریق دستیابی به پنجره ها و مرورگرهایWeb انجام دهند.



شکل ۱۲-۳- محیط کاربری SWAT در Ubuntu

Swat [1] نرم افزار کمکی دستورات و تنظیمات Samba است که از طریق هر مرورگر Web قابل دسترسی است. Swat با دستکاری در فایلهای سیستمیLinux می‌تواند تمامی تنظیمات شبکه شما را ساده تر از آنچه فکر می‌کنید انجام دهد..در اکثر موارد با نصب Samba فایل‌های اجرائی Swat هم نصب می‌شوند. برای مشاهده فایل های مربوطه باید به مسیر/usr/local/samba/bin مراجعه نمایید.

به‌منظور اجرای سرویس Swat به صورت اتوماتیک هنگام بالا آمدن لینوکس ، در مسیر etc/ فایل Services را به کمک نرم افزار Gedit باز کرده و عبارت tcp 901/Swat را در انتهای فایل فوق اضافه می‌کنیم .با انجام این کار، سرویس Swat به صورت اتوماتیک هنگام بالا آمدن لینوکس بر روی درگاه [2] شماره ۹۰۱ از پروتکل TCP اجرا می‌شود.

حال باید Xinetd[3] را برای اوبونتو تنظیم نماییم.برای این منظور به پوشه /etc/xinetd.d رفته و با استفاده از ویرایشگر متن استاندارد در لینوکس فایلی با نام swat ایجا نمایید و دستورات زیر را در آن وارد کنید:

```
#   description: swat is the Samba Web Administration Tool, which
#   allows an administrator to configure Samba using a web
#   browser interface, with the URL http://localhost:901
service swat.
}
    socket_type         = stream
    wait                = no
    protocol            = tcp
    only_from           = localhost
    user                = root
    log_on_failure     += USERID
    server              = /usr/local/samba/bin/swat
    port                = 901
    disable             = no
{
```

بعد از وارد کردن اطلاعات لازم فایل را بسته و آن را ذخیره نمایید.حال برای اجرای سرویس باید یک سیگنال بفرستیم.برای این کار می‌توان از دستور زیر استفاده کرد:

```
#   bin/kill -HUP -a xinetd
```

به این ترتیب می‌توان به وسیله هر مرورگر وب، در هر سیستم عاملی در بستر شبکه TCP با وارد کردن آدرس کارگزار فایل به همراه درگاه ۹۰۱ به تنظیمات کارگزار فایل دسترسی داشت.

---

[1] Samba Web Administration Tools

[2] Port

[3] لینوکس از دو تنظیم به نام های xinetd و inted جهت اجرای سرویس هایی که بعضی اوقات مورد تقاضا قرار می گیرند استفاده می نماید، Internet super Daemons

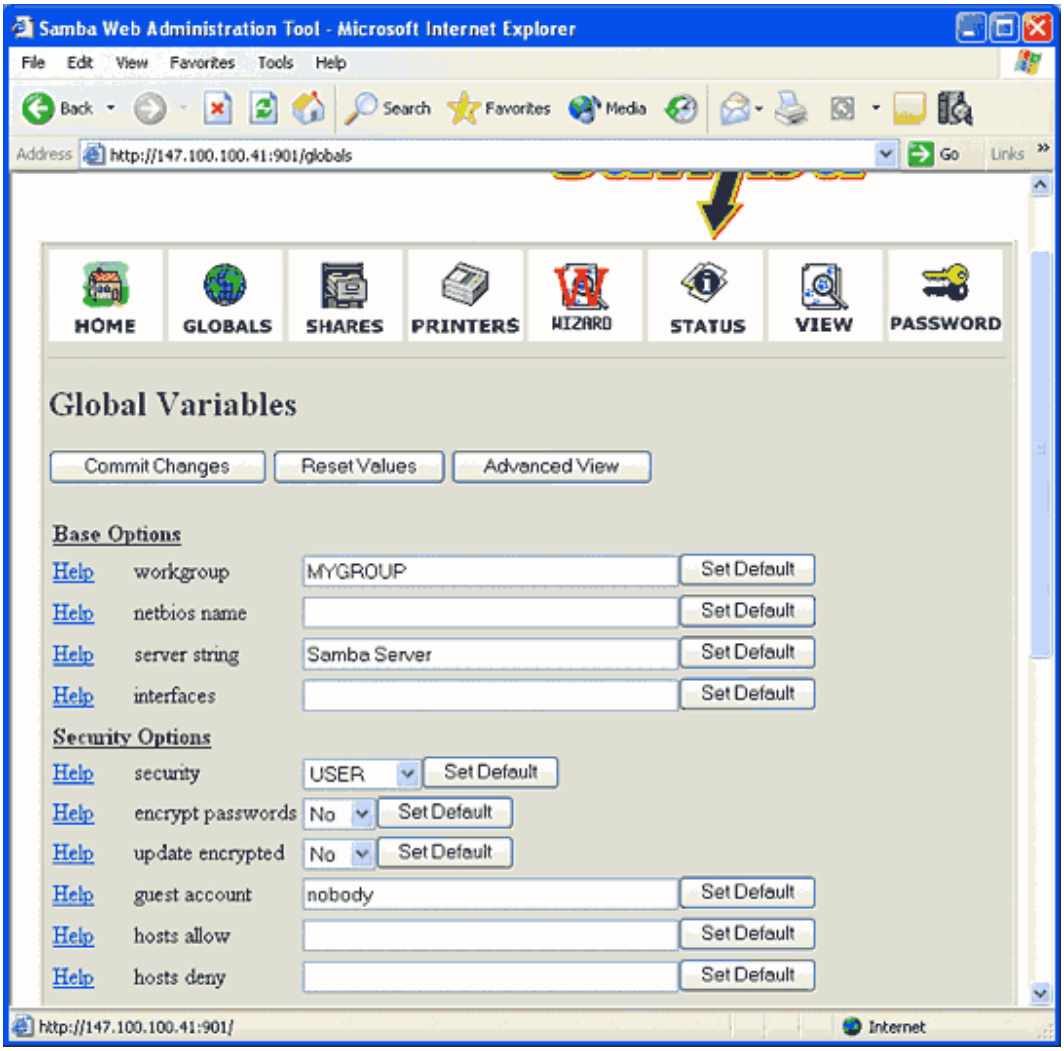
یعنی می توان با وارد کردن آدرسی همانند http://server IP address:901 از خارج کارگزار و با آدرسی همانند http://localhost:901 در داخل کارگزار به تنظیمات Samba دسترسی داشت.البته برای این که به همه تنظیمات و همچنین ایجاد فهرست‌های به اشتراک گذاشته شدهء جدید باید با نام‌کاربری و رمزعبور کاربر ریشه[1] وارد شد.

بعد از ورود به Swat ، راهبر با محیطی کامل روبه‌رو می‌شود که به او امکان هرگونه تنظیمی را که تاکنون از طریق واردکردن دستورالعمل ها انجام می‌داده است را می دهد و اکنون می تواند بسیار آسان‌تر و سریع تر از قبل به کمک ابزارهای گرافیکی که Swat ایجاد کرده است،تنظیمات مورد نظر خود را انجام دهد.

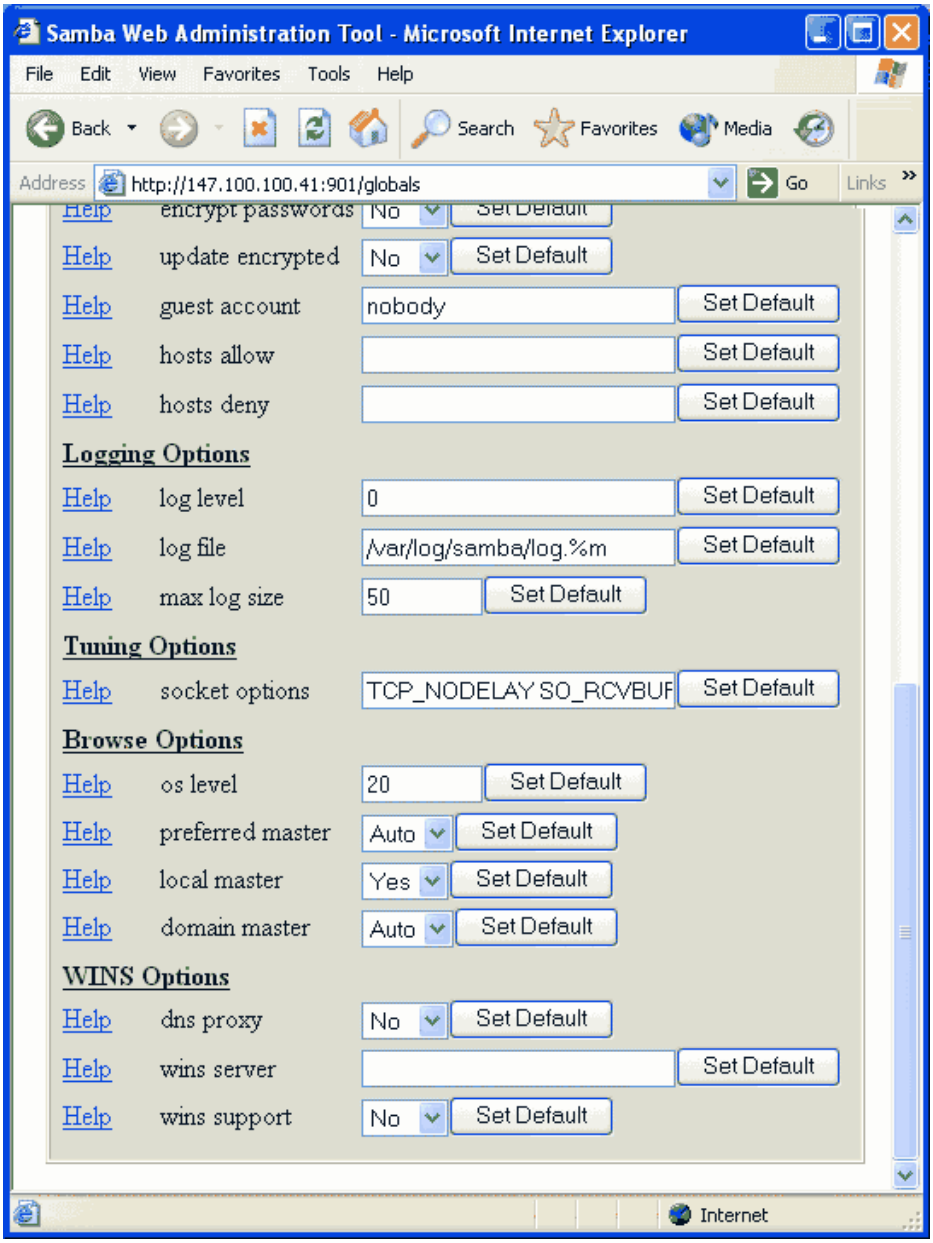
یکی از مهم‌ترین بخش‌های Swat ، بخش Global می‌باشد، که خود شامل تنظیمات مختلفی از جمله تنظیمات پایه‌ای، امنیتی، ورود[2] و میزان سازی[3] که می تواند به همه‌ی فهرست‌های به اشتراک گذاشته شده و خود کارگزار فایل اعمال شود.

- Base Options: این بخش شامل تنظیمات مربوط به کارگزار و TCP/IP می‌باشد.
- Security Options : دو تنظیم امنیتی مهم در این قسمت وجود دارد،یک جهت پنهان‌سازی رمزعبور و دیگری جهت به‌روز رسانی نحوه‌ی پنهان سازی.
- Logging Options: در این بخش متغیری به نام log level قرار دارد که اگر 0 تنظیم شود هیچ کاربری امکان ورود نخواهدداشت، برای این که اجازه ورود به کاربران داده شود، در این قسمت باید عدد 1 وارد گردد.
- Tuning Options : در این بخش تنظیمات مربوط به پروتکل TCP/IP انجام می گیرد.

[1] Root
[2] Logging
[3] Tuning

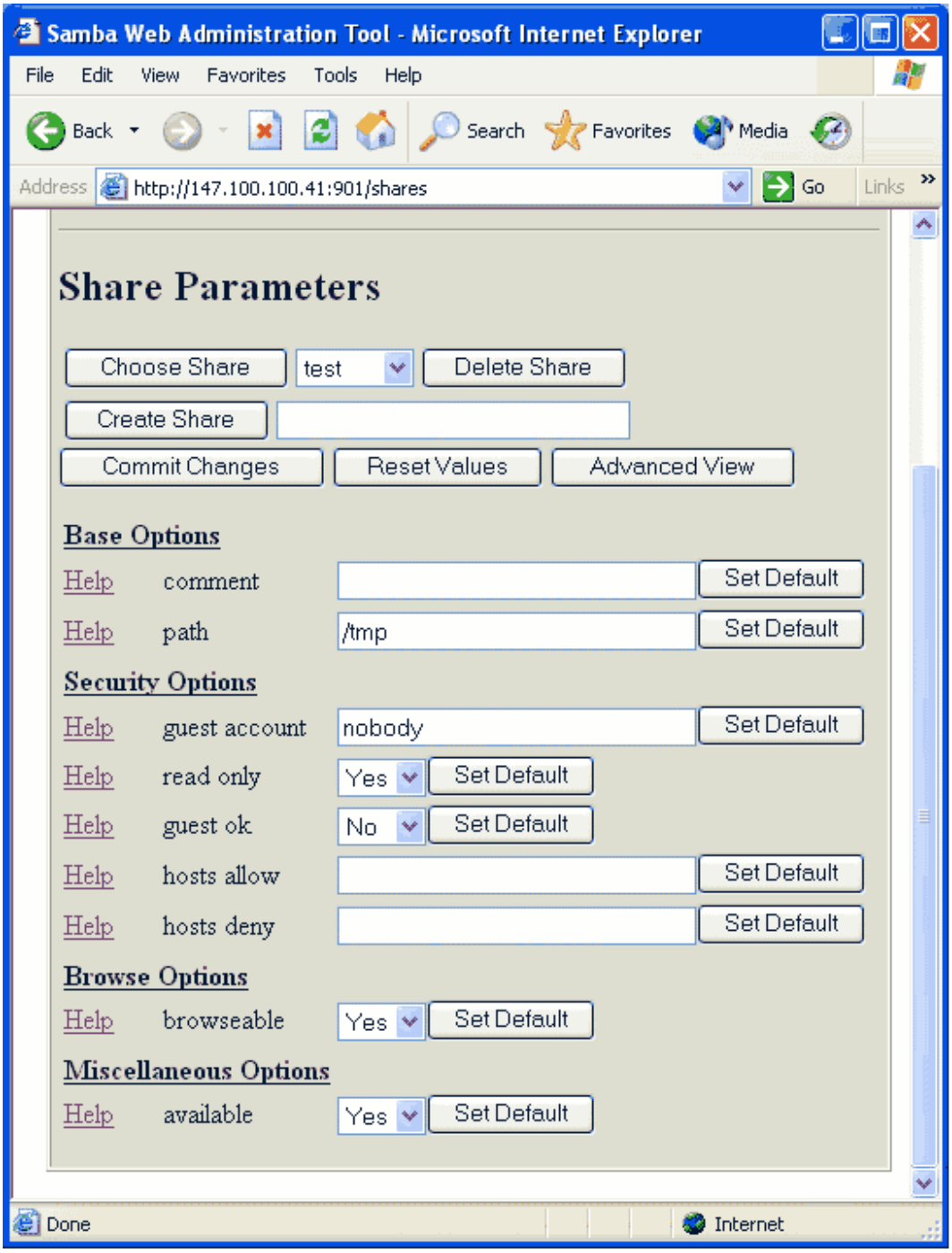


شکل ۱۲-4 محیط SWAT



شکل ۱۲-5 – محیط بخش Global

یکی دیگر از بخش‌های مهم Swat ، بخش Share می‌باشد که با کلیک بر آن به صفحه‌ی مربوط به آن منتقل می شوید.در این بخش راهبر می تواند با ایجاد فهرست‌های مورد نظر خود برای به اشتراک گذاری، هر یک از تنظیماتی را که توسط اعمال دستوراتی در فایل smb.conf انجام می‌داد،در اینجا به صورت گرافیک اعمال نماید.



شکل ۱۲-۶ – بخش Share از Swat

۱-٤- تکلیف جلسه بعد

۱. روش تنظیم Firewall را در لینوکس بیابید و شرح دهید.
۲. در Swat دو بخش دیگر، علاوه بر مواردی که گفته شد، در قسمت Global وجود دارند که یکی Browse Options و دیگری WINS Options می باشد.عملکرد این دوتنظیم را شرح دهید.

۱-۵- مراجع

1. **Using Samba**, 2nd Edition,By Jay Ts, Robert Eckstein, and David Collier-Brown,2nd Edition, February 2003 ,O'Reilly & Associates, ISBN: 0-596-00256-4,www.oreilly.com/catalog/samba2/
2. **Samba-3 by Example Practical Exercises in Successful Samba Deployment,** John H. Samba Team Terpstra ,http://www.samba.org/samba/docs/man/Samba-Guide/
3. **The Official Samba 3.5.x HOWTO and Reference Guide**, http://samba.org/samba/docs/man/Samba-HOWTO-Collection/

۱-٦- دستور کار

روی هر کدام از سیستم‌ها دو نسخه لینوکس Ubuntu (یکی به عنوان لینوکس کارگزار و یکی نیز به عنوان لینوکس کاربر) و یک نسخه ویندوز XP تازه موجود می باشد. همچنین فایل‌های مورد نیاز برای آزمایش نیز روی همه سیستم‌ها به اشتراک گذاشته شده‌است. قبل از شروع آزمایش دو سیستم‌عامل را به وسیله نرم افزار Virtual Box اجرا نمایید.

توجه: بهتر است که یکی از لینوکس‌ها را با نام Server Linux و دیگری را با نام Client Linux در virtual box نام گذاری نمایید تا در طول انجام آزمایش دچار سردرگمی نشوید.

قبل از این که دو سیستم عامل را اجرا نمایید لازم است که از تنظیمات کارت شبکه خود و تنظیمات شبکه Virtual Box اطمینان حاصل کنید.برای انجام درست آزمایش باید کارت شبکه کامپیوترتان حتما فعال باشد.به منظور تنظیم مشخصات شبکه ماشین‌های مجازی ، در Virtual Box وارد گزینه تنظیمات هر کدام از سیستم عامل‌ها شده و در زبانه شبکه، قسمت (Attached to :) را به Host – only adapter تغییر دهید و سپس تایید نمایید.حال شرایط برای شروع آزمایش محیی می‌باشد،حال هر دو سیستم‌عامل را اجرا نمایید.(در این آزمایش از DHCP موجود در خود Virtual Box استفاده می شود ،شماره شبکه پیش فرض ۱۹۲,۱۶۸,۵۶,۰ می‌باشد .در صورتی که مایل به تغییر آن هستید می توانید از مسیر زیر نسبت به تغییر آن اقدام نمایید:

File > Preferences > Network

۱. الف)وارد ویندوز XP خود شوید و با استفاده از دستور ipconfig مشخصات کارت شبکه خود رامشاهده نمایید و آن ها را یادداشت کنید.

System > Preferences >Network Connections

ب) وارد محیط ترمینال در لینوکس شوید و دستور ifconfig را وارد نمایید.خروجی را در گزارش کار خود یادداشت نموده و چند سطر در مورد آن شرح دهید.حدس می زنید که این دستور چه کاربردهایی می‌تواند داشته باشد(برای این منظور می‌توانید از راهنمای لینوکس کمک بگیرید) در ادامه چند مورد از گزینه‌های موجود برای این دستور را به دلخواه شرح دهید.

ج) همان طور که می دانید لینوکس برای هر منظوری از پرونده ها استفاده می‌کند.بنابراین تنظیمات کارت شبکه هم می‌تواند یک پرونده باشد.با استفاده از دستوراتی که در آزمایش‌های پیشین

فراگرفته‌اید سعی کنید که این پرونده را یافته و محتویات آن را مشاهده‌کنید ودر آخر مسیر آن را در گزارشکار خود یادداشت نمایید.

راهنمایی:در این مسیر جستجو نمایید

etc/sysconfig/…

د) با توجه به سوال اول آیا فکر می کنید ویندوز و لینوکس در یک شبکه هستند؟ چرا؟

۲. الف) وارد محیط ویندوز شوید و در خط فرمان آدرس IP لینوکس کارگزار را ping نمایید.خروجی را یادداشت نمایید.

ب) آیا در لینوکس هم، دستور Ping همانند ویندوز استفاده می گردد؟(برای این منظور می‌توانید از راهنمای لینوکس[1] استفاده نمایید.اگر جواب مثبت است ، در محیط ترمینال در لینوکس، ویندوز را Ping نمایید و خروجی را مشاهده نموده و آن را در گزارش کار خود یادداشت نمایید.

ج) با استفاده از راهنمای دستور Ping در لینوکس ، تنظیماتی[2] از این دستور بیابید که به وسیله آن بتوان تعداد بسته های ارسالی را محدود نمود.در صورت وجود چنین دستوری آن را تست و یادداشت نمایید.

به طور متال: Ping -<option> count

توجه: قبل از شروع آزمایش بعد باید از وجود بسته Samba اطمینان حاصل شود برای این کار دستور مقابل را وارد نمایید:

$ smbd –v

اگر بعد از اجرا پاسخی مبنی بر نسخه سرویس نصب‌شده روی لینوکس دریافت نمودید می‌توانید برای انجام بخش بعد اقدام نمایید، در غیر این صورت باید مراحل زیر را جهت نصب بسته Samba طی نمایید.

وارد محیط Virtual Box شده و به مسیر زیر بروید:

Settings>Shared Folders

حال پوشهء Samba utility که روی میزکار کامپیوتر شما قرار دارد و حاوی آخرین بسته Sambaمی‌باشد را به پوشه های به اشتراک گذاشته شده اضافه نمایید.(اطمینان حاصل نمایید که پوشه با دسترسی کامل به اشتراک گذاشته شود)

---

[1] man

[2] Options

هم اکنون دوباره لینوکس خود را اجرا نمایید و برای دستیابی به آن پوشه ، در منوی Places به قسمت Network رفته و پوشه به اشتراک گذاشته شده را بیابید و فایل درون آن را به مسیر etc/samba انتقال دهید و وارد محیط ترمینال شده و مسیر جاری را توسط دستور cd به همان مسیر انتقال بسته تغییر دهید و دستورات زیر را جهت خارج کردن فایل ها از حالت فشرده، اجرا نمایید:

```
$ tar xvfz samba-3.5.2.tar.gz
```

البته می‌توانید با مراجعه به محل پوشه مورد نظر، با راست کلیک روی فایل فشرده و انتخاب گزینه Extract here همین کار را انجام داد.

در این مرحله باید اقدام به کامپایل و نصب سورس[1] Samba نماییم .در این نسخه از Samba ، این فایل در پوشه Source3 قرار دارد، بنابراین باید ابتدا مسیر جاری را به آن پوشه هدایت نماییم . سپسدستورات زیر را به ترتیب انجام دهید:(در طی نصب، لینوکس گزارش فعالیت‌هایی که در حال انجام می‌باشد به شما ارائه می‌شود)

```
$  cd /etc/samba/samba-3.5.2/source3
$  sudo su
#  ./configure
#  make 2>&1 | tee make.log
#  make install
```

جهت اطمینان از نصب سرویس می‌توانید با مراجعه به مسیر /usr/local/samba فایل هلی مربوطه را مشاهده نمایید.

توجه:در صورتی که در طی مراحل نصب با مشکلی برخورد نموده و فرایند نصب ناتمام ماند قبل از شروع دوباره مراحل گفته شده، محیط ترمینال را بسته و بعد از ورود دوباره دستورات زیر را اجرا نمایید:

```
#  autoconf
#  makeclean
#  rm config.cache
```

۳. الف) در نسخه لینوکسی که در دسترس می‌باشد،بسته Samba از قبل موجود می باشد. به آدرس /etc/samba رفته و نگاهی به فایل‌های پیکربندی که در آنجا وجود دارد بیندازید ونام دو عدد از آن ها در گزارش کار خود یادداشت نمایید.

ب) نحوه نصب samba در کارگزار و کلاینت با هم متفاوت می‌باشد.در این بخش از آزمایش در لینوکس کارگزار وارد محیط ترمینال شوید و برای نصب Samba و ایجاد یک Samba Server دستور زیر را تحت کاربر root اجرا نمایید:

---

[1] Source Code

root@PC : apt-get install samba smbclient

ج) در این مرحله روی میزکار لینوکس یک پوشه به نام گروه خود ایجاد کنید. و سپس روی آن کلیک راست کرده و گزینه Sharing option را انتخاب نمایید.سپس در پنجره باز شده ابتدا تیک share this folder را بزنید و همچنین تیک قسمت guest account را جهت دسترسی بدون رمز عبور بردارید..حال وارد ویندوز شوید و با وارد کردن آدرس IP ویا در قسمت My Network Places فایل به اشتراک گذاشته شده را بیابید و به مربی نشان دهید.

۴. الف) با استفاده از دستور زیر به آدرس فایل smb.conf مراجعه نمایید و با استفاده از ویرایشگر Gedit محتویات آن را مشاهده نمایید.این فایل پیکربندی از هفت بخش اصلی تشکیل شده‌است .نام این هفت بخش را در گزارشکار خود بنویسید.

Sudo gedit /etc/samba/smb.conf

ب) حال سعی‌کنید که در این فایل تغییری ایجادنمایید و سپس آن را ذخیره کنید.با چه خطایی مواجه می‌شوید؟برای رفع این مشکل می‌توانید از دستور sudo chown <user> <file path> استفاده کنید .شکل کامل دستوری را که استفاده کرده اید را در گزارش کار خود بنویسید.به نظر شما این دستور چه کاربردی دارد؟

ج) قبل از هر چیز از این فایل یک پشتیبان تهیه‌کرده و در جایی از سیستم نگهداری نمایید تا در صورت خرابی فایل اصلی،وضعیت اولیه قابل بازگشت باشد. برای این منظور می توانید از دستور زیراستفاده کنید:

sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.orginal

د) حال یک Share directory در آدرس /srv/samba/sharegroup با نام گروه خود ایجاد نموده وبا باز کردن این پوشه به اشتراک گذاشته‌شده در ویندوز xp ، نتیجه را به مربی نشان دهید.

ه)حال در این مرحله مشخصات به اشتراک گذاری پوشه‌ای را که در مرحله قبل ایجاد کردید به گونه ای تغییر دهید که امکان نوشتن در این فایل وجود نداشته باشد.(فایل Write Protected باشد)

و) آیا نصب این سرویس به همین شکل امنیت کامل شبکه را تامین می‌نماید؟ به نظر شما چه راهکارهایی برای افزایش امنیت این سرویس می توان به کار برده شود؟

۵. الف) وارد ویندوز شده و پوشه‌ای به نام گروه خود ایجاد نمایید و یک فایل متنی نیز با نام Example.txt و محتوای نام اعضای گروه خود ایجاد کرده و در آن پوشه قرار دهید.سپس پوشه را به روش معمول به اشتراک بگذارید.حال در لینوکس کارگزار به مسیر Place>Network>Windows

Network رفته و فایل به اشتراک گذاشته شده را یافته و به میزکار لینوکس کپی نمایید و نتیجه را به مربی نشان دهید.

ب)این بار برای استفاده از پوشه‌های به اشتراک گذاشته شده در شبکه ویندوز ، پنجره مرورگر موزیلا در لینوکس را باز کرده و در قسمت آدرس، عبارت smb:/// را تایپ و اجرا کنید و فایل Example.txt را در آن مشاهده نمایید.

۶. وارد لینوکس سرویس گیرنده شوید و سرویس Samba Client را در آن به صورت زیر نصب نمایید:

root@PC : apt-get install smbf

با استفاده از دستور smbclient -L //server -U user اشتراک های Samba را لیست نمایید و سپس با دستور smbclient //server/share -U user به اشتراک موجود متصل شویدحال prompt باید به شکل SMB:/> تبدیل شده باشد.(به جای user نام کاربری جاری خود را قرار دهید)

الف) با استفاده از دستور Help لیست کامل دستورات قابل استفاده را بیابید و سه مورد از دستورات آشنا را به دلخواه توضیح دهید. چه دستوراتی از آن بادستورات FTP مشترک می‌باشند.

ب) فایلی با نام smbclient.txt در لینوکس کارگزار به اشتراک بگذارید و با استفاده از این دستورات فایل را به لینوکس سرویس‌گیرنده خود انتقال دهید.نتیجه را به مربی نشان دهید.

۷. در لینوکس کارگزار، خدمت SWAT را به ترتیبی که در پیش آگاهی آمده است فعال نمایید و سپس وارد لینوکس کلاینت شده و با استفاده از مرورگر موزیلا وارد Swat شده و نتیجه را به مربی نمایش دهید.سپس با ورود به سربرگ Share یک فهرست به نام SwatShare ایجاد نموده و آن را به اشتراک بگذارید و با استفاده از تنظیمات آن، آن را طوری تنظیم نمایید که فقط قابل خواندن باشد.

فصل سیزدهم

گزارش کار آزمایش‌ها

۱۳-۱- مقدمه

همانطور که پیش از انجام هر آزمایش بایستی، پیش‌آگاهی را به دقت مطالعه نمود. پس از انجام هر آزمایش باید یک گزارش کار به مربی تحویل داده شود. پیشنهاد می‌شود دانشجویان تلاش نمایند تا پرسش‌های مطرح شده در دستورکار آزمایش را با همفکری اعضای گروه حل نمایند و در حد امکان، به مربی و سایر گروه‌ها ارجاع ندهند.

همچنین از پرسش‌ها و آزمایش‌های کلیدی گزارش کار ممکن است برای آزمون نهایی نظری و عملی، استفاده شود، بنابراین پیشنهاد می‌شود نکات کلیدی در این زمینه را برای برای خود یادداشت کنید.

در ادامه گزارش‌کارهای آزمایش‌های مطرح شده در این مستند ضمیمه شده است تا در حین آزمایش، پر شود و به مربی تحویل داده شود. لازم به یادآوری است که تحویل تنها یک گزارش به ازای هرگروه کفایت می‌کند.

گزارش‌کار باید در همان جلسه‌ی آزمایشگاه نوشته و به مربی تحویل داده شود و خارج کردن آن از آزمایشگاه به دلیل شائبه‌ی استفاده‌ی گروه‌های بعدی، مجاز نیست.