آزمایش چهارم

برنامه‌نویسی پوسته لینوکس

۱,۱- 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.

۲,۱- هدف

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

---

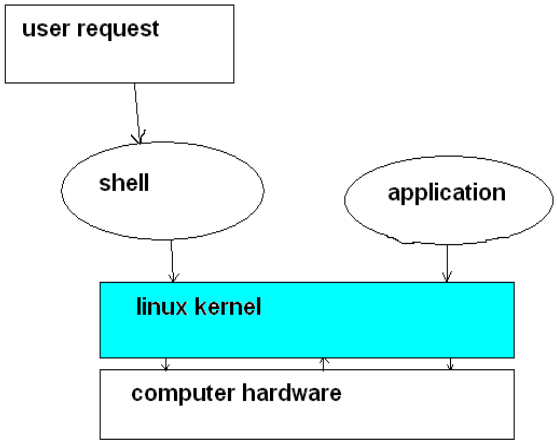
[1] Shell
[2] Back up

۳,۱- 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.

۱,۳,۱- 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 ۱- نقش هسته در سیستم‌عامل لینوکس

هسته 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[3]

۵- مدیریت حافظه

[3] Files

۲,۳,۱- پوسته[4] 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 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[5] 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.

---

[4] Shell
[5] Syntax

۳,۳,۱- 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.

٤,١- 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.

١,٤,١- 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

۲,٤,۱- 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[6] UDV که به‌وسیله‌ی

[6] User Defined Variable

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.

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

$ ver=""

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

٦. 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]

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 ۱-۱- 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"**

۳,٤,۱- 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[۷] 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 دستور if

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.

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

[7] Exit Status

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 ۲-۱- 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.

۵,۱- 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 continueرا 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 می‌گیرد. نکته‌ی دیگر اینکه می‌توان به جای عبارت [ ] در شرط حلقه، از دستور test نیز استفاده کرد.

٦,١- مراجع

1. Mark Sobell, Matthew Helmke, **Practical Guide to Linux Commands, Editors, and Shell Programming**, 4’th Ed., Prentice-Hall Inc., 2017.

۷,۱- 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 عمکلرد 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. 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
```