۲- مقدمه این جلسه

در جلسات پیش با طریقه کار با پوسته ترمینال آشنا شدیم.همچنین چند دستور مفید لینوکس را یاد گرفته ایم.شما می توانید دستورات ترمینال خود را در یک فایل قرار داده،و این فایل را فراخوانی کرده،تا تمامی دستورهای داخل آن اجرا شود.به آن فایل یک Script و به کاری که شما انجام داده اید،Shell Scripting می گویند.البته این مبحث،مبحث پیچیده ای است،و فقط این نیست که من چند دستور در یک فایل قرار دهم،و آن ها را اجرا کنم.امکاناتی که در Shell Scripting نهاده شده است،امکاناتی است که شاید قابل مقایسه با یک زبان برنامه نویسی کامل باشد.پس از یادگیری این مبحث،با چند دستور مدیریتی مهم ترمینال لینوکس آشنا خواهیم شد.

۳- SHELL SCRIPTING چیست؟

دستورهای shell که داخل یه فایل می‌نویسیم تا پشت سر هم اجرا بشن (مثل فایل‌های Bat. و Cmd. در MS Win/DOS)

یک فایل با نام test.sh با استفاده از محیط گرافیکی گنوم ساخته،و سپس دستورات زیر را در آن ذخیره کنید:

```
echo "Hello World!"
echo "Salam be jahan!"
```

حال می توانید با تایپ دستور ./test.sh این فایل را اجرا کرده،و خروجی آن را ببینید. (البته اگر در مسیری باشید که این فایل در آنجا باشد،در غیر این صورت باید با دستور cd به آن مسیر بروید،یا اینکه آدرس کامل فایل را تایپ کنید.)

نکته: هر فایلی که اجرا می کنید،باید دسترسی اجرای آن را داشته باشید.اگر ندارید،می توانید از دستور chmod به این صورت استفاده کنید:

chmod +x نام فایل

همچنین می توانید در محیط گرافیکی گنوم،بر روی فایل کلیک راست کرده،گزینه Properties را زده،و در قسمت Permissions تیک گزینه Allow executing file as Program را بزنید.

۴- نوشتن COMMENTدر برنامه

با گذاشتن علامت # در ابتدای هر خط ، آن خط از برنامه comment می شود.

۵- پایان خط در SHELL

اول بگم که هر جا خواستیم به خط جاری پایان بدیم و ادامه‌ی اون رو توی خط بعدی بنویسیم از \ استفاده می‌کنیم:

```
echo \
"Salam"
```

دوم هر کجا که به خط بعد رفتیم معادل ; حساب می‌شه و بلعکس. مثلا دستورات

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

با دستور زیر معادل است:

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

۶- متغیرها

متغیرها به صورت name=value مقداردهی می‌شوند،و با $name فراخوانی می‌شوند.

```
myname="Amir"
echo "My name is $myname"
```

۷- دستور READ

با دستور read می‌تونیم از ورودی متغیرها رو بخونیم.

```
read varname
```

۸- بلوک IF

ساختار if به صورت زیر است:

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

به جای condition شرط را وارد می‌کنیم و جای commands دستورات رو. قسمت‌های elif و else اختیاری هستند برای نوشتن conditionهای مختلف از دستور test استفاده می‌کنیم،به دو صورت می‌تونیم از test استفاده کنیم:

- test condition
- [condition]

مثال:

```
read name

if [ $name  =  "Amir" ]
then
      echo "Your name is Amir"
elif [ $name  =  "Mehdi" ]
then
      echo "Your name is Mehdi"
else
      echo "I dont know you"
fi
```

shell شرط ورودی رو اجرا می‌کنه و اگر مقدار صفر رو برگردوند دستورات رو اجرا می‌کنه و اگر غیر صفر بود اجرا نمی‌کنه (یا else رو اجرا می‌کنه)

بهتره این‌جوری بگم که برعکس زبان‌هایی مانند خانواده‌ی C که صفر برابر False (غلط) و غیر صفر برابر True (درست) هست توی shell صفر برابر True و غیر صفر برابر False هست

۹- پارامترهای مختلف TEST

دستور test پارامترهای زیادی داره بعضی از اون‌ها رو نوشتم.در زیر کلمه exp به معنای یک عبارت،str به معنای یک رشته،int به معنای یک عدد صحیح،و file به معنای یک فایل می باشد.

- **!exp** : اگر exp غلط باشد (not)
- **exp1 -a exp2** : اگر exp1 و exp2 هر دو درست باشند (and)
- **exp1 -o exp2** : اگر exp1 یا exp2 یا هردو درست باشند (or)
- **str** : اگر طول str ناصفر باشد
- **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 یاشد
- **-d file** : اگر file یک پوشه باشد
- **-f file** : اگر file وجود داشته باشد و یک فایل معمولی باشد

برای اطلاعات بیشتر به man test مراجعه کنید

۱۰- حلقه ها

۱۰-۱- بلوک WHILE

```
while condition
do commands
done
```

تا وقتی که condition درست باشد (کد خروجی از اجرای condition صفر باشد) commands اجرا می‌شوند

مثال: script زیر هر ثانیه یک x رو صفحه می‌گذارد و وقتی تعداد آن‌ها به ۵ رسید حلقه‌ی while به پایان می‌رسه

```
a=
while [ "$a" != "....." ]
do
	a= "$a."
	echo x
	sleep 1
done
```

نکته: دستور Sleep برای متوقف کردن فرآیند اجرای برای به مدت یک ثانیه به کار رفته است.

به جای while != condition می‌توانیم از until condition استفاده کنیم

```
a=
until [ "$a" = "....." ]
do
	a = "$a."
	echo x
	sleep 1
done
```

۱۰-۲- بلوک FOR

```
for name in list
do commands
done
```

به ازای هر بار اجرای command متغیر name یکی از مقادیر list رو به خود می‌گیره

مثال: این script اعداد ۱ تا ۶ رو می‌نویسه

```
for number in 1 2 3 4 5 6
do
	echo -n "$number "
done
echo
```

نکته: گزینه n– در دستور echo باعث می شود که این دستور خروجی را در یک خط نمایش دهد،و به خطوط دیگر نرود.دستور echo آخر نیز برای رفتن به خط جدید است.

مثال: این یکی اسم تمام فایل‌های پوشه جاری رو می‌نویسه

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

۱۰-۳- دستورهای BREAK و CONTINUE

با دستور continue دستورهای بعدی حلقه اجرا نمی‌شود و دوباره به اول حلقه می‌رود.با دستور break دستورهای بعدی حلقه استفاده اجرا نمی‌شود و برنامه از حلقه خارج می‌شود

اگر حلقه‌های تو در تو داریم و می‌خواهیم دو یا چند بار break یا continue کنیم جلوی این دستورات تعداد را وارد می‌کنیم

مثال: این script تمام اعداد دو رقمی قبل از ۷۳ را می‌نویسد

```
for a in 1 2 3 4 5 6 7 8 9
do
	for b in 0 1 2 3 4 5 6 7 8 9
	do
		echo "$a$b"
		if [ "$a$b" = "72" ]
		then
			break 2
		fi
	done
done
```

۱۰-۴- بلوک CASE

```
case name in
match1) commands ;;
match2) commands ;;
esac
```

اگر می‌خواهیم مقدارهای مختلف یک متغیر را بررسی کنیم می‌توانیم به جای چندین if از case استفاده کنیم در صورتی که می‌خواهیم چند مقدار را بررسی کنیم (با یک عبارت) آن‌ها را با | از هم دیگر جدا می‌کنیم مثلا:

```
name1|name2)
        commands
        ;;
```

مثال: این Script از ورودی یک عبارت می گیرد.اگر برابر listFiles باشد،دستور ls را برای نمایش فایل های اجرا می کند.اگر برابر showDirectory باشد،دستور pwd را برای نمایش پوشه فعلی اجرا می کند.همچنین اگر عبارت ورودی برابر Calendar باشد،تاریخ و ماه فعلی رو چاپ می کند،اگر هیچ کدام از این موارد نباشد،پیغام خطا می دهد.

```
echo -n "Enter your command: "
read cmd

case $cmd in
   listFiles)
      ls ;;

  showDirectory)
      pwd ;;

   calendar)
      date
      echo
      cal
      ;;
     *)
      echo "Ooops! Commands are: listFiles, showDirectory, calendar" ;;
esac
```

همانطور که در مثال بالا دیدیم،اگر هیچ کدام از مقادیر ارائه شده درست نباشد،بخش (*) از دستور case اجرا می شود.

۱۱-علامت های || و &&

این عبارت را در نظر بگیرید:

```
command1 && command2
```

این دستور command1 را اجرا می‌کند و اگر خروجی صفر بود (درست) command2 اجرا می‌شود،مثلا کد زیر را در نظر بگیرید:

```
[ -f fileName ] && ./fileName
```

این قطعه کد (که خیلی استفاده می‌شه) یعنی اگر فایل fileName وجود داشت آن را اجرا کن در غیر این صورت کاری انجام نمی‌دهد.

نکته: اگر بخواهیم چند کار انجام دهیم آن‌ها رو با {} احاطه می‌کنیم

نکته: || برعکس && هست یعنی اگر خروجی command1 غیر صفر بود command2 اجرا می‌شود

۱۲- تعریف توابع

```
FunctionName()
{
    commands
    .
    .
    .
    return exitStatus
}
```

exitStatus یک عدد صحیح است که به عنوان خروجی در نظر گرفته می‌شود.برای فراخوانی تابع از اون جایی تابع همون برنامه است، از همون syntax فراخوانی برنامه ها استفاده می‌کنیم

func param1 param2

و کد خروجی هم (از اون‌جایی که یه برنامه هست) توی متغیر محیطی $? ذخیره می‌شه.

۱۳- آراگومان ها

در جلسات قبل دیدیم که هر دستور ممکن است آراگومان و گزینه هایی داشته باشد.اگر بخواهیم آراگومان هایی که کاربر برای اجرای Script ما استفاده کرده است،را بخوانیم،چه کار باید بکنیم؟این کار،بسیار ساده است.

آراگومان اول توی $1، آرگومان دوم توی $2، ... و آراگومان صفرم توی $0 ذخیره می‌شه.مثلا اگر یک Script با نام test.sh حاوی کد زیر بسازید:

```
echo "0 = $0 1 = $1 2 = $2"
```

و سپس دستور زیر را برای اجرای این Script وارد کنید:

```
./test.sh hello world
```

آنگاه خروجی به این صورت است:

```
0 = ./test.sh 1 = hello 2 = world
```

۱۳-۱- آراگومان های خاص

- آراگومان صفرم ($0) همون اسم برنامه هست (به همون طریقی که اجرا شده)
- متغیر $# تعداد آراگومان ها را مشخص می‌کنه
- متغیر $@ شامل تمام متغیرها است که وقتی به string تبدیل می‌شود آراگومان ها را جداگانه مشخص می‌کند
- متغیر $* شامل تمام متغیرها است که وقتی به string تبدیل می‌شود آراگومان ها به هم می‌چسبند و بین آن‌ها اولین کاراکتر متغیر IFS هست (که معمولا space هست)
- متغیر $$ شماره پروسه shell جاری هست
- متغیر $! شماره پروسه آخرین دستور پیش زمینه یا آخرین پایپ(علامت |) است

مثال: Script زیر را با دستور ./test.sh "hello world" hello2 world2 اجرا کرده،تا بفهمید هر آراگومان خاص چه خروجی دارد.

```
echo "0 = $0"
echo "# = $#"
echo "@ = $@"
for a in "$@"; do
  echo $a
done
IFS="/"
echo "* = $*"
for a in "$*"; do
  echo $a
done
echo "\$ = $$"
```

خروجی آن در کامپیوتر بنده به این صورت است:

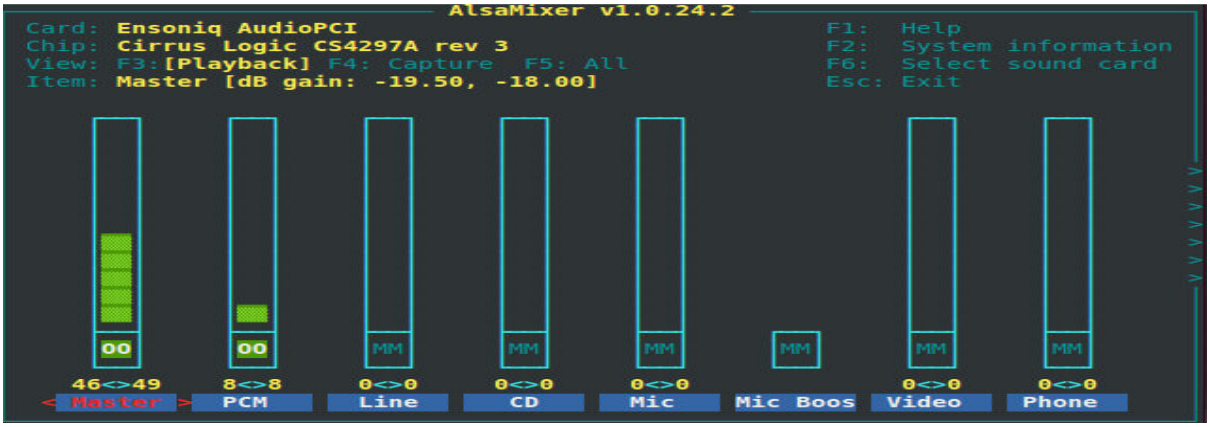
```
0 = ./test.sh
# = 3
@ = hello world hello2 world2
hello world
hello2
world2
* = hello world/hello2/world2
hello world hello2 world2
$ = 19750
```

۱۴- دستورات مدیریتی در لینوکس

در این قسمت،چند دستور مدیریتی که برای کار با سخت افزار می باشد،ارائه شده است.

- **alsamixer** : برای تنظیم صدای Speaker کامپیوتر است.با اجرای این برنامه،می توانید تنظیمات صدا را در یک محیط گرافیکی اعمال کنید.(کلیدهای بالا و پایین برای زیاد و کم کردن صدا هستند.از کلید Esc برای خروج استفاده کنید.)

تصویر این برنامه در شکل ۱ نشان داده شده است.



شکل ۱: دستور alsamixer

- **lspci :** با این دستور می توانید لیستی از سخت افزارهای نصب شده روی سیستم را ببینید.
- **ifconfig :** با این دستور می توانید سخت افزارهای مربوط به شبکه را تنظیم کنید.مثلا اگر بخواهید به کارت شبکه ای به نام eth0 یک IP به آدرس 192.168.0.10 و یک subnet mask به صورت 255.255.0.0 بدهید،از دستور زیر استفاده کنید:

Ifconfig eth0 192.168.0.10 netmask 255.255.0.0 up

نکته: گزینه up در آخر دستور بالا برای این است،که در صورت خاموش بودن کارت شبکه،این کارت فعال شود.از گزینه down برای خاموش کردن کارت شبکه استفاده می شود. **نکته:** اگر می خواهید تنظیمات فعلی کارت شبکه را ببینید، دستور ifconfig را بدون آراگومان وارد کنید.

نکته: هر تنظیمی که با استفاده از دستور ifconfig انجام می دهید،بعد از ریست شدن سیستم از بین می رود.

- **cfdisk** : این دستور،برنامه ای را اجرا می کند،که با آن می توانید هارد دیسک خود را پارتیشن بندی کنید.
- **startx** : این دستور باعث می شود محیط گرافیکی لینوکس(گنوم) اجرا شود.اگر در پوسته فرمان لینوکس قرار دارید،و می خواهید وارد محیط گرافیکی شوید،از این دستور استفاده کنید.

تمرین ها

۱- همانطور که متوجه شده اید،برای نشان دادن یک رشته در Script،باید آن را در داخل علامت دابل کتیشن قرار داد.مثلا باید بنویسیم:

```
echo "this is a test"
```

اما می توان رشته ها را در داخل سینگل کتیشن نیز گذاشت،یعنی نوشت:

```
echo 'this is a test'
```

تفاوت بین استفاده از دابل کتیشن برای نمایش رشته،و سینگل کتشین چیست؟

۲- در این دستور کار،مثال زیر را ارائه دادیم:

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

گفتیم این Script می تواند فایل های پوشه جاری را نمایش دهد.اما نگفتیم که آن علامت * چه کاربردی دارد.این علامت معنای خاصی در Shell Scripting می دهد.این معنا چیست؟و دیگر چه علامت هایی وجود دارند که مانند این معنای خاصی می دهند؟

۳- در هنگام معرفی دستور ifconfig گفتیم: " هر تنظیمی که با استفاده از دستور ifconfig انجام می دهید،بعد از ریست شدن سیستم از بین می رود."

حال اگر بخواهیم **بدون استفاده از محیط گرافیکی**،IP کارت شبکه را عوض کنیم،بدون اینکه این IP بعد از ریست شدن سیستم از بین برود،چه کار باید بکنیم؟

۴- در این جلسه فرصت نشد تا راجع به مدیریت فایل در اسکریپت نویسی بحث شود.تحقیق کنید چگونه می توان یک Script نوشت که در آن Script یک دستور(مثلا ls) فراخوانی شود،و **به جای اینکه** خروجی این دستور در صفحه نمایش نشان داده شود،در یک فایل ذخیره شود.سپس یک Script بنویسید که دستور ls را اجرا کند،و خروجی آن را در یک فایل ذخیره کند.

۵- شما می توانید در Script های خود از عبارات ریاضی نیز استفاده کنید،مثلا دستورات زیر ابتدا مقدار ۲ ضربدر ۸ در متغیر a قرار می دهد.سپس این متغیر را با ۴ جمع کرده،و در انتها خروجی(که مساوی ۲۰ است) را نمایش می دهد.

```
let "a=2*8"
let "a=a+4"
echo $a
```

عملگرهای ریاضی شامل موارد روبرو است: ضرب(*)،جمع(+)،تفریق(-)، تقسیم (/)، و باقیمانده(%).

یک Script بنویسید که دو عدد به صورت پارامتر از ورودی بگیرد،و مقدار ترکیب آن ها را حساب کنید.

راهنمایی: می دانیم فرمول ترکیب به صورت روبرو است: $\binom{a}{b} = \frac{a!}{b!(a-b)!}$ پس باید در Script خود یک تابع برای محاسبه فاکتوریل ساخته،و سپس از آن برای محاسبه ترکیب استفاده کنید.

فهرست منابع و ماخذ این جلسه

۱. **اردوخانی، آرمان**. اسکریپت نویسی در محیط شل. [درون خطی] http://wiki.ubuntu.ir/ShellScripting.

۲. **مهاجرانی، امید**. پیکربندی کارت شبکه با ifconfig. [درون خطی] http://linux-notes.blogfa.com/post-4.aspx.

3. **Thomas, Keir and Channelle, Andy.** *Beginning Ubuntu Linux 11.04.* s.l. : APress.