



## آزمایشگاه سیستم عامل دستور کار ۷: Bash Script

Bash مفسر زبان دستورات است. shell یک پردازشگر ماکرو است که دستورات را اجرا می کند. اصطلاح پردازشگر ماکرو به معنای عملکردی است که در آن متن و نمادها برای ایجاد عبارت بزرگتر گسترش می یابد.

Unix Shell مترجم دستورات و یک زبان برنامه نویسی است. به عنوان یک مفسر دستور، Shell رابط کاربری مشتمل بر مجموعه ای غنی از سرویس های GNU ارائه می دهد. ویژگی های زبان برنامه نویسی اجازه می دهد که این سرویس ها ترکیب شوند. فایل های که حاوی دستورات هستند می توانند ایجاد شوند و خودشان تبدیل به دستور شوند. این دستورات جدید همانند دستورات سیستم در دایرکتوری هایی مانند `/bin` هستند، این امکان را ایجاد می کنند تا کاربران یا گروه ها محیط هایی شخصی را برای بهینه سازی کارهای معمول خود ایجاد کنند.

با نوشتن مجموعه ای از دستورات در یک فایل متنی می توان به جای اجرای تک تک دستورات در ترمینال لینوکس با اجرای فایل به این هدف دست یافت. اگر قالب این فایل به صورت `sh` باشد و اسم فایل را `sample.sh` فرض شود، می توان با دستور `./sample.sh` این فایل را اجرا کرد. حالت دیگر آن است که از دستور `bash filename` در ترمینال استفاده شود. برای ایجاد این فایل لازم است در اولین خط عبارت `#!/bin/bash` را قرار گیرد تا مشخص شود مفسر این دستورات `bash` است.

### مقداردهی متغیرها

مانند هر زبان برنامه نویسی دیگر در این زبان هم می توان متغیر تعریف کرد و به آن مقادیری نسبت داد. بدین منظور به مثال های زیر دقت کنید برای چاپ مقادیر از دستور `echo` استفاده می شود.

```
#!/bin/bash
#variable assignment
# no space around = during assignment
a=24
echo $a
echo "$a"
echo "The value of \"a\" is $a."
a=`echo Hello!` # Assigns result of 'echo' command to 'a' ...
echo $a
a=`ls -l`      # Assigns result of 'ls -l' command to 'a'
echo "$a"
echo $a        # Unquoted, however, it removes tabs and newlines.

# Assignment using 'let'
let a=16+5
echo "The value of a is now $a."
```

متغیرهای خاصی وجود دارند که مقادیر آن‌ها از قبل تعیین شده‌اند و می‌توان در کاربردهای خاص از آن‌ها استفاده کرد. مانند:

**\$0 – \$1 - \$9 – \$# – @\$ – \$\$ – \$USER –**

- **\$0** - The name of the Bash script.
- **\$1 - \$9** - The first 9 arguments to the Bash script. (As mentioned above.)
- **#** - How many arguments were passed to the Bash script.
- **@** - All the arguments supplied to the Bash script.
- **?** - The exit status of the most recently run process.
- **\$** - The process ID of the current script.
- **\$USER** - The username of the user running the script.
- **\$HOSTNAME** - The hostname of the machine the script is running on.
- **\$SECONDS** - The number of seconds since the script was started.
- **\$RANDOM** - Returns a different random number each time it is referred to.
- **\$LINENO** - Returns the current line number in the Bash script.

در هنگام اجرای فایل می‌توان ورودی داد مثال:

```
bash samplefile 1 3
```

که در آن مقادیر ۱ و ۳ که با فاصله آمده‌اند آرگومان هستند و برای استفاده از این آرگومان‌ها باید از متغیرهای \$1 و \$2 استفاده کرد.

برای دریافت مقادیر مورد نیاز از کاربر در حین اجرای برنامه از دستور **read** استفاده می‌شود.

```
read -p 'Username: ' uservar
read -sp 'Password: ' passvar
```

برای انجام محاسبات شیوه‌های مختلفی وجود دارد. به مثال‌های زیر توجه کنید.

```

let a=10+8
echo $a
expr 5 \* 4
expr 5 / 4
expr 11 % 2
a=$( expr 10 - 3 )
echo $a
b=$(( a + 3 ))
echo $b
((b++))
echo $b

```

عبارات شرطی:

برای نوشتن شرط از قالب زیر پیروی کنید:

```

if [ ] then
elif [ ] then
else
fi

```

اگر چند شرط مختلف داشته باشیم می توان این گونه آن ها را استفاده کرد: `[]&&[] []||[]`

برای مقایسه اعداد می توان از `-eq -gt -lt` استفاده کرد که در مثال زیر به کار رفته است:

```

var1=10
var2=20
if [ $var1 -gt $var2 ] then
    echo "$var1 is greater than $var2"
fi

```

عبارات چند حالتی:

قالب دستور case:

```

case $variable in
    pattern-1)
        commands
        ;;
    pattern-2)
        commands
        ;;
    pattern-3|pattern-4|pattern-5)
        commands
        ;;
    pattern-N)
        commands
        ;;
    *)
        commands
        ;;
esac

```

### حلقه‌ها:

قالب حلقه while و مثالی از آن در ادامه آمده است:

```
while [ condition ]
do
    command1
    command2
    command3
done

counter=0
while [ $ counter -lt 10 ]
do
    echo The counter is $ counter
    let counter = counter +1
done
```

قالب حلقه for و مثالی از آن در ادامه آمده است:

```
for VARIABLE in 1 2 3 4 5 .. N
do
    command1
    command2
    command3
done

for VARIABLE in file1 file2 file3
do
    command1
    command2
    command3
done

for OUTPUT in $(Linux-Or-Unix-Command-Here)
do
    command1
    command2
    command3
done

for i in $( ls )
do
    echo item: $i
done
```

مثال:

```
#!/bin/bash
for i in 1 2 3 4 5
do
    echo "Welcome $i times"
done
```

## توابع

تعریف توابع به صورت زیر تعریف می‌شوند:

```
function_name() {
    command1
    command2
    command3
    #return
}
```

دقت شود قبل از تعریف تابع نمی‌توان از آن استفاده کرد. اگر در تابع از دستور `return` استفاده گردد مقدار آن توسط `$?` قابل دسترسی است. برای ارسال آرگومان به تابع مشابه برنامه عمل می‌شود. به مثال زیر دقت کنید:

```
#!/bin/bash
# Setting a return status for a function
print_something () {
    echo Hello $1
    return 5
}
print_something Mars
print_something Jupiter
echo The previous function has a return value of $?
```

## تمرین:

- ۱- اسکریپتی بنویسید که دو عددی که به صورت آرگومان به آن داده شده را:
  - الف) با هم جمع کند.
  - ب) عدد بزرگتر را نشان دهد.
  - ج) میانگین اعداد بین دو عدد ورودی را چاپ کند.
- ۲- با استفاده از `case` ماشین حسابی را با دو ورودی طراحی کنید.
- ۳- اسکریپتی بنویسید که رمز عبور کاربر را از آن بگیرد. در صورتی که رمز عبور وارد شده با عبارت "۱۲۳۴۵۶" معادل باشد، پیام `Welcome` و در غیر این صورت پیام `Wrong Password` را نمایش دهد.