Operating Systems

UE22CS242B

Suresh Jamadagni
Department of Computer Science and Engineering

Operating Systems

Bash shell and cron

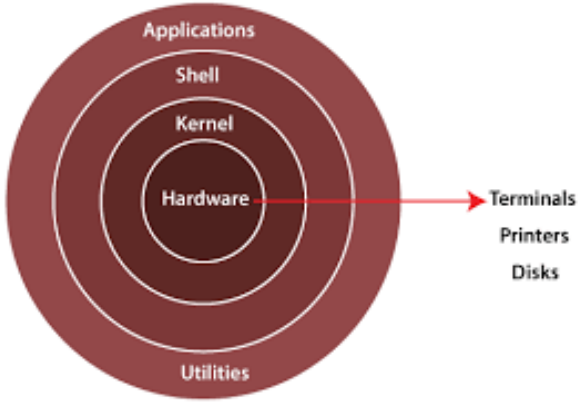
Suresh Jamadagni

Department of Computer Science and Engineering

OPERATING SYSTEMS

What is a shell?

- The **shell** is the Linux command line interpreter.
- Shell provides an interface between the user and the kernel and executes programs called commands.
- For example, if a user enters ***ls*** then the shell executes the *ls* command.
- The shell can also execute other programs such as applications, scripts, and user programs



OPERATING SYSTEMS

What is a shell?

- Instructions entered in response to the shell prompt have the following syntax: **command [arg1] [arg2] .. [argn]**
- The brackets [] indicate that the arguments are optional. Many commands can be executed with or without arguments
- The shell parses the words or tokens (commandname , options, filesnames[s]) and gets the kernel to execute the commands assuming the syntax is correct.
- Typically, the shell processes the complete line after a carriage return (cr) (carriage return) is entered and finds the program that the command line wants executing.
- The shell looks for the command to execute either in the specified directory if given (./mycommand) or it searches through a list of directories depending on your $PATH variable.

OPERATING SYSTEMS

Various shells

- There are a number of shells available to a Linux user:

  sh - This is known as the Borne Shell and is the original shell

  csh, tcsh - These are well-known and widely used derivatives of the Borne shell

  ksh - The popular Korn shell

  bash - The Borne Again SHell is the most popular shell used for linux and developed by GNU

echo $SHELL

```
ubuntu@ip-172-31-41-208:~$ echo $SHELL
/bin/bash
```

OPERATING SYSTEMS

Environment variables

- **Environmental variables** are variables that are defined for the current shell and are inherited by any child shells or processes.
- Environmental variables are used to pass information into processes that are spawned from the shell.
- **Shell variables** are variables that are contained exclusively within the shell in which they were set or defined.
- They are often used to keep track of ephemeral data, like the current working directory.
- By convention, these types of variables are usually defined using all capital letters. This helps users distinguish environmental variables within other contexts.
- We can see a list of all of our environmental variables by using the ***env*** or ***printenv***

OPERATING SYSTEMS

Common Environment variables

- **SHELL**: This describes the shell that will be interpreting any commands you type in. In most cases, this will be bash by default, but other values can be set if you prefer other options.
- **TERM**: This specifies the type of terminal to emulate when running the shell. Different hardware terminals can be emulated for different operating requirements. You usually won't need to worry about this though.
- **USER**: The current logged in user.
- **PWD**: The current working directory.
- **OLDPWD**: The previous working directory. This is kept by the shell in order to switch back to your previous directory by running cd -.
- **PATH**: A list of directories that the system will check when looking for commands. When a user types in a command, the system will check directories in this order for the executable.
- **HOME**: The current user's home directory.

OPERATING SYSTEMS

Common SHELL variables

- **BASHOPTS**: The list of options that were used when bash was executed. This can be useful for finding out if the shell environment will operate in the way you want it to.
- **BASH_VERSION**: The version of bash being executed, in human-readable form.
- **BASH_VERSINFO**: The version of bash, in machine-readable output.

```
ubuntu@ip-172-31-41-208:~$ echo $BASH_VERSION
5.0.17(1)-release
```

OPERATING SYSTEMS

SHELL basics

- Creating shell variables:

```
ubuntu@ip-172-31-41-208:~$ MY_VAR="Hello world"
```

- Accessing the value of any shell or environmental variable:

```
ubuntu@ip-172-31-41-208:~$ echo $MY_VAR
Hello world
```

- Spawning a child shell process

```
ubuntu@ip-172-31-41-208:~$ bash
```

- In the child shell process, the shell variable defined in the parent is not available

```
ubuntu@ip-172-31-41-208:~$ echo $MY_VAR
```

- To terminate the child shell process,

```
ubuntu@ip-172-31-41-208:~$ exit
```

- To pass the shell variables to child shell processes, you need to **export** the variable

```
ubuntu@ip-172-31-41-208:~$ export MY_VAR="Hello world"
ubuntu@ip-172-31-41-208:~$ bash
ubuntu@ip-172-31-41-208:~$ echo $MY_VAR
Hello world
```

PES UNIVERSITY
CELEBRATING 50 YEARS

OPERATING SYSTEMS

SHELL basics

- Removing a shell variable

```
ubuntu@ip-172-31-41-208:~$ unset MY_VAR
ubuntu@ip-172-31-41-208:~$ echo $MY_VAR

ubuntu@ip-172-31-41-208:~$
```

- For setting environment variables at login, edit the **.profile** file in the **$HOME** directory and add the export command

```
ubuntu@ip-172-31-41-208:~$ cd $HOME
ubuntu@ip-172-31-41-208:~$ vi .profile
```

OPERATING SYSTEMS

SHELL control flow - if

```
if commands; then
    commands
[elif commands; then
    commands...]
[else
    commands]
fi
```

Example: To check if the file exists

```
#!/bin/bash

if [ -f welcome.txt ]; then
   echo "File exists"
else
   echo "File does not exist"
fi
```

OPERATING SYSTEMS

SHELL control flow - loops

```
#!/bin/bash

for i in {1..5}
do
   echo "i = $i"
done
```

Nested loop:

```
#!/bin/bash

for i in {1..5}
do

   for j in {1..3}
   do
      echo "i = $i and j = $j"
   done
done
```

OPERATING SYSTEMS

SHELL control flow – continue and break

```
#!/bin/bash

for i in {1..5}
do

   if [ $i -eq 3 ]
   then
      continue
   fi

   for j in {1..3}
   do
      echo "i = $i and j = $j"
   done
done
```

```
#!/bin/bash

for i in {1..5}
do

   if [ $i -eq 3 ]
   then
      break
   fi

   for j in {1..3}
   do
      echo "i = $i and j = $j"
   done
done
```

OPERATING SYSTEMS

cron

- In Unix and Linux, **cron** is a daemon, which is an unattended program that runs continuously in the background and wakes up (executes) to handle periodic service requests when required. The daemon runs when the system boots
- Cron is a job scheduling utility present in Unix like systems. The **crond** daemon enables cron functionality and runs in background.
- The cron reads the **crontab** (cron tables) for running predefined scripts.
- By using a specific syntax, you can configure a cron job to schedule scripts or other commands to run automatically.
- Checking for jobs scheduled in **cron**

```
ubuntu@ip-172-31-41-208:~$ crontab -l
no crontab for ubuntu
```

OPERATING SYSTEMS

cron

- Starting/checking status of **cron**

```
ubuntu@ip-172-31-41-208:~$ sudo systemctl status cron.service
● cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2024-01-24 06:44:21 UTC; 1h 8min ago
```

- Adding jobs to the scheduler
- **crontab -e**: edits crontab entries to add, delete, or edit cron jobs.

```
*   *   *   *   *  sh /path/to/script/script.sh
|   |   |   |   |             |
|   |   |   |   |      Command or Script to Execute
|   |   |   |   |
|   |   |   |   |
|   |   |   |   |
|   |   |   | Day of the Week(0-6)
|   |   |   |
|   |   | Month of the Year(1-12)
|   |   |
|   | Day of the Month(1-31)
|   |
| Hour(0-23)
|
Min(0-59)
```

OPERATING SYSTEMS

cron examples

SCHEDULE	SCHEDULED VALUE
5 0 * 8 *	At 00:05 in August.
5 4 * * 6	At 04:05 on Saturday.
0 22 * * 1-5	At 22:00 on every day-of-week from Monday through Friday.

PES UNIVERSITY
CELEBRATING 50 YEARS

OPERATING SYSTEMS

cron examples

Executing a program every minute

crontab -l

```
# m h  dom mon dow   command
*/1 * * * * /bin/bash $HOME/OS/hello.exe &> $HOME/OS/cron_output.log
```

```
ubuntu@ip-172-31-41-208:~/OS$ tail /var/log/syslog
Jan 24 08:38:01 ip-172-31-41-208 CRON[2923]: (ubuntu) CMD ($HOME/OS/hello.exe)
Jan 24 08:38:01 ip-172-31-41-208 CRON[2920]: (CRON) info (No MTA installed, discarding output)
Jan 24 08:39:01 ip-172-31-41-208 CRON[2927]: (ubuntu) CMD ($HOME/OS/hello.exe)
Jan 24 08:39:01 ip-172-31-41-208 CRON[2926]: (CRON) info (No MTA installed, discarding output)
Jan 24 08:39:09 ip-172-31-41-208 crontab[2911]: (ubuntu) REPLACE (ubuntu)
Jan 24 08:39:09 ip-172-31-41-208 crontab[2911]: (ubuntu) END EDIT (ubuntu)
Jan 24 08:39:41 ip-172-31-41-208 crontab[2932]: (ubuntu) LIST (ubuntu)
Jan 24 08:40:01 ip-172-31-41-208 cron[665]: (ubuntu) RELOAD (crontabs/ubuntu)
Jan 24 08:40:01 ip-172-31-41-208 CRON[2935]: (ubuntu) CMD (/bin/bash $HOME/OS/hello.exe &> $HOME/OS/cron_output.log)
Jan 24 08:41:01 ip-172-31-41-208 CRON[2939]: (ubuntu) CMD (/bin/bash $HOME/OS/hello.exe &> $HOME/OS/cron_output.log)
```

THANK YOU

Suresh Jamadagni
Department of Computer Science and Engineering

sureshjamadagni@pes.edu