Chapter Table of Contents

Chapter 3.2

Process Control

Aim

To understand file systems of Linux and functions of super user

Instructional Objectives

After completing this chapter, you should be able to:

- Explain different types of file system available in Linux
- Describe how to create a new file system after partitioning of hard disk
- Illustrate mounting and unmounting file system
- Explain the process of checking and monitoring system performance
- Discuss file security and permission in detail
- Describe how to become a super user
- Illustrate how to extract system information using uname command
- Explain how to install and remove packages using rpm command

Learning Outcomes

At the end of this chapter, you are expected to:

- Mount a file in the file system
- Monitor system performance through various open source tools
- Identify the security enabled features of UNIX systems
- Use crypt command to encrypt file
- Use uname command to retrieve system information
- Install and remove packages using rpm command

3.2.1 Introduction

Whenever a program starts its execution, a process is born and as soon as the program terminates, the process dies. The kernel is responsible for management of process. It is responsible for creation and termination of a process. Time and resources reserved for a process, handle priorities when multiple processes are executing at the same time and sharing the CPU.

As files are identified by inode, a process is identified as a unique ID (i.e. PID). Attributes of a process are maintained by kernel using a process table. The important attributes of a process are:

- Process-id (PID)
- Parent-PID(PPID)

3.2.2 Process Creation

A new process can be created using fork system call. The mechanism of creating a process is divided in three phases that uses three system calls: fork, exec and wait. The three phases of process creation are discussed below:

Fork():

It is the only system call that is used to create a new process. The process that calls fork is called the parent process and the new created process is known as child process. The new created process image is identical to parent process except for a few parameters like PID. Following is the syntax of fork system call:

pid= fork();

As a result of fork system call, a new child process is created. Here in the parent process PID is the PID of child process. Process with PID 0 is created when the system is booted. It is the only process that is created without using fork.

Following are the sequence of operations that are performed for fork process:

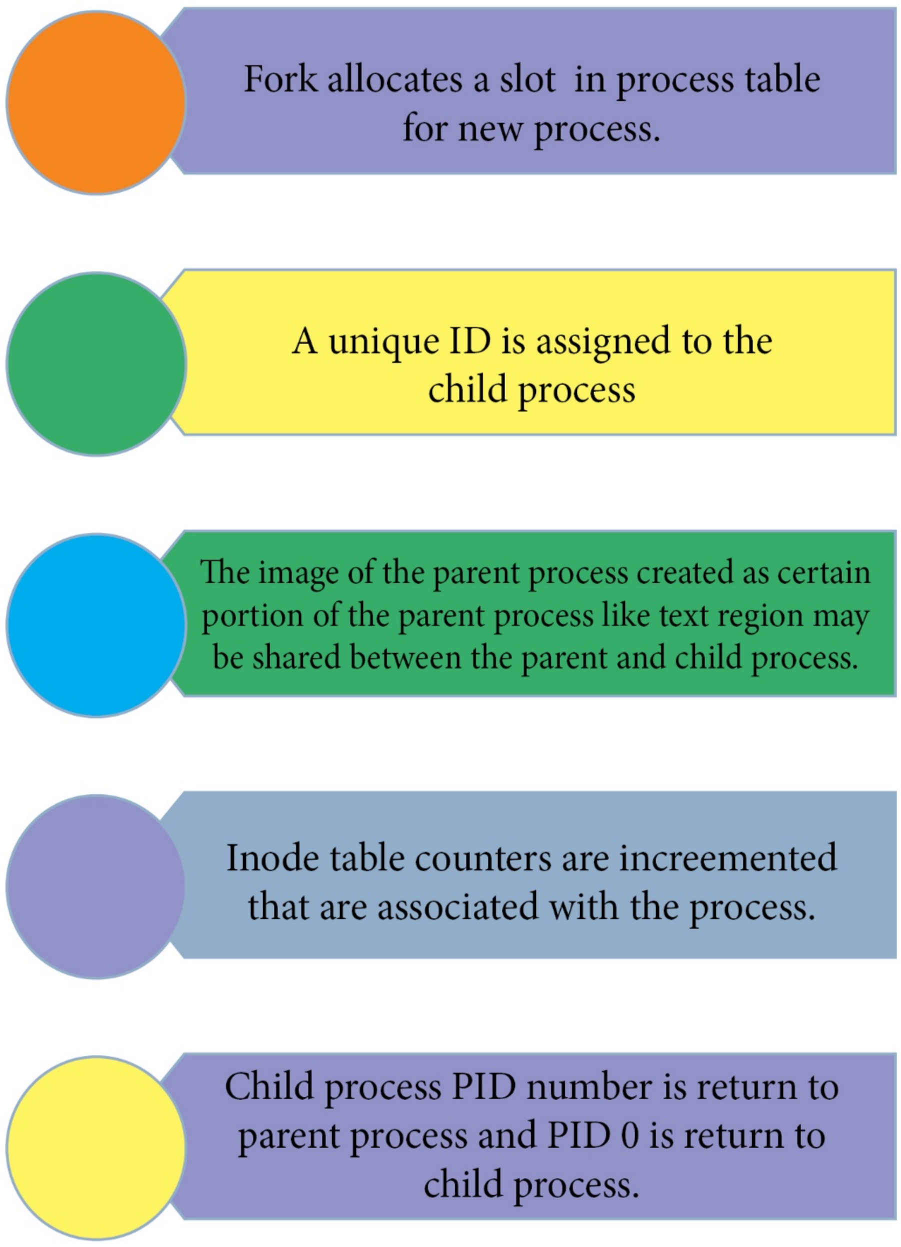
Fork allocates a slot in process table for new process.
A unique ID is assigned to the child process
The image of the parent process created as certain portion of the parent process like text region may be shared between the parent and child process.
Inode table counters are increemented that are associated with the process.
Child process PID number is return to parent process and PID 0 is return to child process.

Figure 3.2.1

Exec():

By using fork system call a new process created but it is not enough to run a new program. To run a new program under fork child it is required to overwrite its own image with code and data of new program. This can be achieved by using EXEC system call. This call does not create a new process and PPID of the executed process remains unchanged. As the result of this system call the child process executes a new program, rather than just duplicating the task of parent process.

Wait():

After using EXEC system call that allow execution of a new program in child process the parent waits for the child process to complete. For this purpose, wait system call is used by the parent process. Wait system call catches the exact status of the child process and then continues its function.

For example: When you run a cat command from the shell, the shell as a parent process forks another child process. The child process with the help of execute system call overlays itself the image of cat and start running the cat command. The parent process i.e. Shell process waits for the cat process to terminate and picks up the exit status of cat process.

Self-assessment Questions

1) In Unix, Which system call creates the new process?
   a) Fork b) Create
   c) New d) None of the mentioned

2) Which system call returns the process identifier of a terminated child?
   a) Wait b) Exit
   c) Fork d) Get

3) A new process is created in Linux by copying the
   a) Address b) ID
   c) Object d) Attributes

4) In Linux address space defines virtual address space assigned to
   a) Processor b) Process
   c) Memory d) Virtual Memory

5) A fork system call will fail. If
   a) The previously executed statement is also a fork call
   b) The limit of maximum processes in system has exceeded
   c) The limit of maximum processes in system that can be executed under single user has exceeded
   d) Bothe (b) and (c)

3.2.3 Signals

A The UNIX system requires to communicate the occurrence of the event to a process. A Signal is a mechanism used by the kernel to communicate the occurrence of the event to a process. Generally, a process responds when it terminates, however there are other actions that a process can take. The action that a signal will take is known as it's disposition. A signal is identified by an integer and its symbolic name. Generally, signal name should be used, rather than its number. You can see name-number translation map in <singal.h>. Signals can be generated by events, take place in hardware, OS or elsewhere.

For example: The signal SIGINT is generated by pressing [ctrl c], this signal terminates a process. Similarly, signal SIGTSTP uses [ctrl z]. Signal can be generated as a result of arithmetic exception, illegal instructions etc.

Signal disposition:

If a program receives a signal, there are following three things it can do:

-**Ignore the signal**: In this case the process continues its execution and completely ignore the signal. Example: when the SIGCHLD signal is send to parent process when a child dies, the parent process ignores the signal.

-**Terminate the process**: Some signals receive by the process causes the process to terminate. It is default dispositions of many signals. Signals generated by Hardware errors, also terminate the processes.

-**Stop the process:** This signal SIGSTOP invokes when you process [ctrl z] from the keyboard.

The default action of a signal can be caught. A signal is generated and delivered to a process. A signal is delivered only when the signal disposition has occurred. When a signal sent to a process, the kernel sets a bit in the pending signal mask field of the process table to indicate that a signal of a specific type has been received by the process. This field has a bit reserved for each type of signal. The process checks this field and the signal disposition table. Accordingly, the process decides to ignore, terminate or invoke a signal handler.

The kernel gets cautious when the process is in the midst of execution of a system call. If the process is sleeping on the completion of a disk I/O, the kernel allows the system call to complete before performing signal delivery. However, if the process is waiting to take input from the terminal, the kernel will abort the call and wake up the process. That is reasonable because waiting for terminal input could be wait forever.

Self-assessment Questions

6) The action that a signal will take is known as its ____________

a) Dislocation	b) Disposition
c) Relocation	d) None of these

7) The signal SIGINT is generated by pressing _______________

a) [ctrl d]	b) [ctrl z]
c) [ctrl c]	d) [ctrl s]e

8) The signal SIGTSTP is generated by pressing ________________

a) [ctrl c]	b) [ctrl d]
c) [ctrl s]	d) [ctrl z]

3.2.4 Process termination

A process is terminated by using exit system call. The exit system call return the status to the parent process. A process can terminate in any of the following ways:

- A process terminates at the end of a program where you have not used an explicit exit or return call.
- By using return statement explicitly in main program.
- By using exit function or exit system call anywhere in the program.
- On receiving a signal that can terminate a process.

Following is the syntax of exit system call:

Void exit (int status);

When a process terminates, kernel sends a signal (SIGCHLD) to parent process to inform the death of child process. Exit status of child process is collected by parent process by invoking wait or waitpid system call.

When a parent process fork a child, a new duplicate image of parent process is created. Child process can use exec system call to execute some other program. Hence, either a child process uses exec system call or not, the two processes (i.e. parent and child process) run independently. When a child process is under execution, the parent process can do two things:

- Waiting for child's exit status
- Continue execution of parent process without waiting for child process to complete execution

When a child process terminates, its exit status is kept in process table. The parent process collects this status by using wait system call. The wait system call blocks the parent process until the child process dies. When a parent process spawn multiple child processes, the wait returns the status as soon as the first child process dies. Therefore, wait system call can't handle process groups. To overcome these problems waitpid is used.

Self-assessment Questions

9) A process is terminated by using ________ system call

a) Exec	b) Fork
c) Exit	d) Wait

10) The exit system call return the status to the ________ process.

a) Child	b) Parent
c) Parent and child	d) None of these

11) A process can be terminated due to ________________

a) Normal exit	b) Fatal error
c) Killed by other process	d) All of the mentioned

3.2.5 PID and PPID

PID:

When a process is created, kernel allocates a unique integer to a process which is known as process id (I.e. PID). This PID is used by kernel to control a process.

To know the PID of your current shell, you can type the following command:

```
$ echo $$
291
```

Knowledge of PID is important to control a process.

PPID:

Like files, each process has a parent. Parent is also a process. A process born from a parent process is known as child process. The PID of the parent process is one of the important process attribute. If many processes share same PID, in that case rather than deleting all the child processes, you can kill the parent process only.

When you type a command cat on shell as following:

```
$ cat abc.txt
```

A process representing cat command starts in shell process. Here, shell process is the parent process and cat is the child process.

Every process has a parent and every process can have only one parent. The ancestor of every process is first process with PID 0, that is set up at the time when system is booted. It is like root directory of file system.

3.2.6 Invoking other programs

The fork system call creates a duplicate image of parent process. Generally, the purpose of forking is to run a separate program in a forked process. This can be accomplished using exec as discussed previously in this chapter.

How exec work and its comparison does with fork is explained in following figure 3.2.1

Fork ()	Exec()
Kernel replicate the address space of the parent process (i.e. text, data, stack, etc.).	Kernel replaces the entire address space (i.e. text, data, stack, etc.) of child process with that of new program.
At this point, there are two processes with same constituents and both continue execution after fork.	
Forking is responsible for creating processes	But , its exec system call that actually executes a program on UNIX system

Let us see the usage of fork and exec system calls through following example code:

```
Main()
{

        int status;
        if ( fork() == 0 )
                execl(" /bin/date", "date", 0);
                wait(& status);
}
```

Here the fork system call created a child process by replicating its address space. Then the child process invokes the exec system call. As a result of exec system call, parent and child process execute independent copies of the program.

3.2.7 Sell on a shell

- Categorize the types of a shell
- Explain the process to execute the shell script

The shell provides a user an environment where he can run commands, programs and shell scripts. The shell is a command interpreter .there are different types of shells. Each shell has its own set of commands and functions.

There are two major shells:

The Bourne shell: the default prompt is $. It is UNIX's first shell environment. It was written by Stephen R. Bourne in mid 1970s. the default term "shell" is used for referring to Bourne shell. It is the most preferred shell for writing scripts.

The C shell: the default prompt is %.

Subcategories for Bourne shell and C shell are given below:

Bourne shell	C shell
Bourne shell (sh) Korn shell (ksh) Bourne Again shell (bash) POSIX shell (sh)	C shell (csh) TENEX/TOPS C shell (tcsh)

In this chapter, we are going to discuss shell script on Bourne shell.

Shell Scripting:

Sometimes it is required to execute a group of commands. This group of commands should be stored in a file and this file can be executed as shell script. Shell scripts use .sh extension which is not mandatory.

A shell script can include conditional tests, loops, instructions to read and store data, and variables to read and store data etc. A shell script can also include functions.

Shell scripts and functions are both interpreted. This means they are not compiled.

Let us create a shell script:

Note:

- Shell script would use .sh extension.
- Before writing any scripting statement, you need to tell the system that a shell script is being started by using #!/bin/sh in the first line of the script which is interpreter line. "/bin/sh" is the pathname of the shell used to execute the script.
- "/bin/sh" is where Bourne shell is located in UNIX.
- You can put comments in the script by putting # tag in front of comments.

Put the following instructions in your script:

```
#!/bin/sh

pwd

ls
```

Now save the contents as " test.sh".

Now shell script can be executed using following instruction:

```
$ . /test.sh
```

Self-assessment Questions

12) In System call __________, Kernel replicate the address space of the parent process to child process

a) Exec	b) Fork
c) Wait	d) Waitpid

13) In System call __________, Kernel replaces the entire address space of child process with that of new program.

a) Exec	b) Fork
c) Wait	d) Waitpid

14) The shell provides a user an environment where he can run___________________.

a) Commands	b) Programs
c) Shell scripts	d) All of these

15) Default prompt of Bourne shell is___________.

a) %	b) $
c) #	d) !

16) Default prompt of C shell is___________.

a) %	b) $
c) #	d) !

Summary

- Whenever a program starts its execution, a process is born and as soon as the program terminates, the process dies.
- The kernel is responsible for management of process. It is responsible for creation and termination of a process. Time and resources reserved for a process, handle priorities when multiple processes are executing at the same time and sharing the CPU.
- The important attributes of a process are Process-id (PID) and Parent-PID(PPID).
- When a process is created, kernel allocates a unique integer to a process which is known as process id (I.e. PID). This PID is used by kernel to control a process.
- Every process has a parent and every process can have only one parent. The ancestor of every process is first process with PID 0, that is set up at the time when system is booted. It is like root directory of file system.
- A new process can be created using fork system call. The new created process image is identical to parent process except for a few parameters like PID.
- To run a new program under fork child it is required to overwrite its own image with code and data of new program. This can be achieved by using EXEC system call.
- The UNIX system requires to communicate the occurrence of the event to a process. A Signal is a mechanism used by the kernel to communicate the occurrence of the event to a process.
- A process is terminated by using exit system call. The exit system call return the status to the parent process.
- The shell provides a user an environment where he can run commands, programs and shell scripts. The shell is a command interpreter .there are different types of shells. Each shell has its own set of commands and functions.
- Sometimes it is required to execute a group of commands. This group of commands should be stored in a file and this file can be executed as shell script. Shell scripts use .sh extension which is not mandatory.

- A shell script can include conditional tests, loops, instructions to read and store data, and variables to read and store data etc. A shell script can also include functions.

Terminal Questions

1. Explain the phases to create a new process in the UNIX OS.
2. Explain how to check and handle signals in the process state diagram.
3. Define PID and PPID and their importance.
4. Categorize different types of a shell.

Answer Keys

Self-assessment Questions	
Question No.	Answer
1	a
2	a
3	d
4	b
5	d
6	b
7	c
8	d
9	c
10	b
11	d
12	b
13	a
14	d
15	b
16	a

Activity

Activity Type: Offline **Duration: 30 Minutes**

Description:

1) Rearrange the sequence of operations for fork.

   a) It makes a logical copy of the context of the parent process. Since certain portion of a process, such as the text region, may be shared between processes, the kernel can sometimes increase a region reference count instead of copying the region to a new physical location in memory.

   b) It allocates a slot in the process table for the new process.

   c) It increments file and inode table counters for file associated with the process.

   d) It assigns a unique ID number to the child process.

   e) It returns the ID number of the child to the parent process, and a 0 value to the child process.

Bibliography

e-References

- This website was referred on 3rd May 2016 while developing topic on Process control http://www.cim.mcgill.ca/~franco/OpSys-304-427/lecture-notes/node16.html
- This website was referred on 3rd May 2016 while developing topic on Process control http://www.cs.vu.nl/~ast/books/mos2/sample-10.pdf
- This website was referred on 3rd May 2016 while developing topic on Process control http://www.ee.surrey.ac.uk/Teaching/Unix/unixintro.html
- This website was referred on 3rd May 2016 while developing topic on Process control http://www.linuxnix.com/run-shell-script-linux/

External Resources

- Maurice J. Bach, The Design of Unix Operating System, (2010) Pearson Education
- S. Prata, Advance UNIX, a Programmer's Guide, (2011), BPB Publications, and New Delhi,
- B.W. Kernighan & R. Pike, The UNIX Programming Environment, (2009) Prentice Hall of India.
- Jack Dent Tony Gaddis, Guide to UNIX Using LINUX, (2010) Vikas/ Thomson Pub. House Pvt. Ltd.

Video Links

Topic	Link
The Linux File System	https://www.youtube.com/watch?v=2qQTXp4rBEE
Directory structure of the UNIX file system	https://www.youtube.com/watch?v=PEmi550E7zw

Notes: