MODULE - III

Unix Process Management

MODULE 3

Unix Process Management

Module Description

In this module, we shall discuss about various attributes of a process. Data structures which are used to save the process status are also discussed in this chapter. To master, the techniques of process control, we will use system calls related to the process creation, execution and process termination. In this module, we will also discuss what the role of signals in process management is and execution of shell scripts is also discussed.

By the end of this module, you will have an overview of kernel's role in process management. You can examine the process attributes. Knowledge of process creation will help you to write programs that create processes. At the end of this module you would be able to create and execute processes.

Chapter Table of Contents

Chapter 3.1

Structure of Process

Aim

To understand process management techniques

Instructional Objectives

After completing this chapter, you should be able to:

- Discuss, process state model for the UNIX system and set of state transitions
- Describe the principles of memory management for processes and kernel
- Explain how the operating system and hardware cooperate to do virtual memory translation
- Explain the components of the context of a process

Learning Outcomes

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

- Explain process state transition diagram
- Differentiate between Kernel's virtual address space and physical address space
- Describe how the process state change from running to terminated
- Illustrate the method of changing mode from user to kernel
- List the components of register context

3.1.1 Introduction

In the last two modules, we had a brief overview of system structure and file system. Now it is time to dig deeply into kernel and look more closely to the concept of UNIX process management. Processes are used by operating system to effectively and efficiently manage various system activities. Every time you run a command on UNIX system, a process is initiated. A program under execution is known as a process. Let us have a more detailed discussion on processes in Linux.

3.1.2 Process States and Transitions

UNIX is a multiuser, multitasking and timesharing system. It means, in UNIX system many programs can run at the same time i.e. many processes can be present at the same time but each process takes its turn for running. Therefore, multiple processes wait for CPU for execution. When a process is pre-empted for the execution of some other process, the state of earlier process must be saved so that it can resume its execution later.

Two kernel data structures are used to define the state of a process:

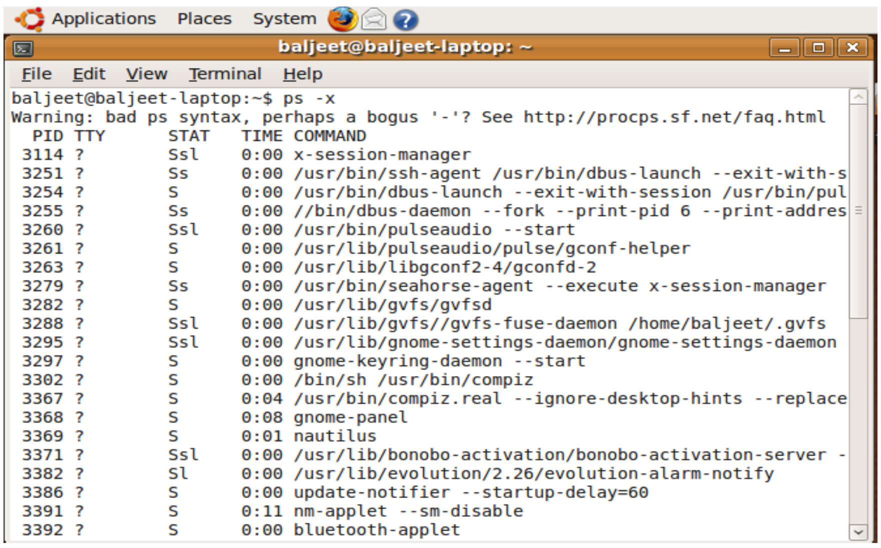
- Process Table
- Uarea

Process table contains different fields. State field, specifies in which state a process is. Process table entry helps the kernel to locate the process and its uarea. This table also contains the information about various user identifiers and process identifiers (UIDs and PIDs). Process table also contains scheduling parameters and timers.

Uarea has a pointer to process table.

If you want to see the currently running processes on your system, you can use ps command with –x option as follows

```
$ ps -x
```



```
baljeet@baljeet-laptop:~$ ps -x
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html
  PID TTY      STAT   TIME COMMAND
 3114 ?        Ssl    0:00 x-session-manager
 3251 ?        Ss     0:00 /usr/bin/ssh-agent /usr/bin/dbus-launch --exit-with-s
 3254 ?        S      0:00 /usr/bin/dbus-launch --exit-with-session /usr/bin/pul
 3255 ?        Ss     0:00 //bin/dbus-daemon --fork --print-pid 6 --print-addres
 3260 ?        Ssl    0:00 /usr/bin/pulseaudio --start
 3261 ?        S      0:00 /usr/lib/pulseaudio/pulse/gconf-helper
 3263 ?        S      0:00 /usr/lib/libgconf2-4/gconfd-2
 3279 ?        Ss     0:00 /usr/bin/seahorse-agent --execute x-session-manager
 3282 ?        S      0:00 /usr/lib/gvfs/gvfsd
 3288 ?        Ssl    0:00 /usr/lib/gvfs//gvfs-fuse-daemon /home/baljeet/.gvfs
 3295 ?        Ssl    0:00 /usr/lib/gnome-settings-daemon/gnome-settings-daemon
 3297 ?        S      0:00 gnome-keyring-daemon --start
 3302 ?        S      0:00 /bin/sh /usr/bin/compiz
 3367 ?        S      0:04 /usr/bin/compiz.real --ignore-desktop-hints --replace
 3368 ?        S      0:08 gnome-panel
 3369 ?        S      0:01 nautilus
 3371 ?        Ssl    0:00 /usr/lib/bonobo-activation/bonobo-activation-server -
 3382 ?        Sl     0:00 /usr/lib/evolution/2.26/evolution-alarm-notify
 3386 ?        S      0:00 update-notifier --startup-delay=60
 3391 ?        S      0:11 nm-applet --sm-disable
 3392 ?        S      0:00 bluetooth-applet
```

Figure 3.1.1

Here in above snapshot you can see each process has a unique id PID (i.e. process ID)

A UNIX process has following states as shown in figure 3.1.1:

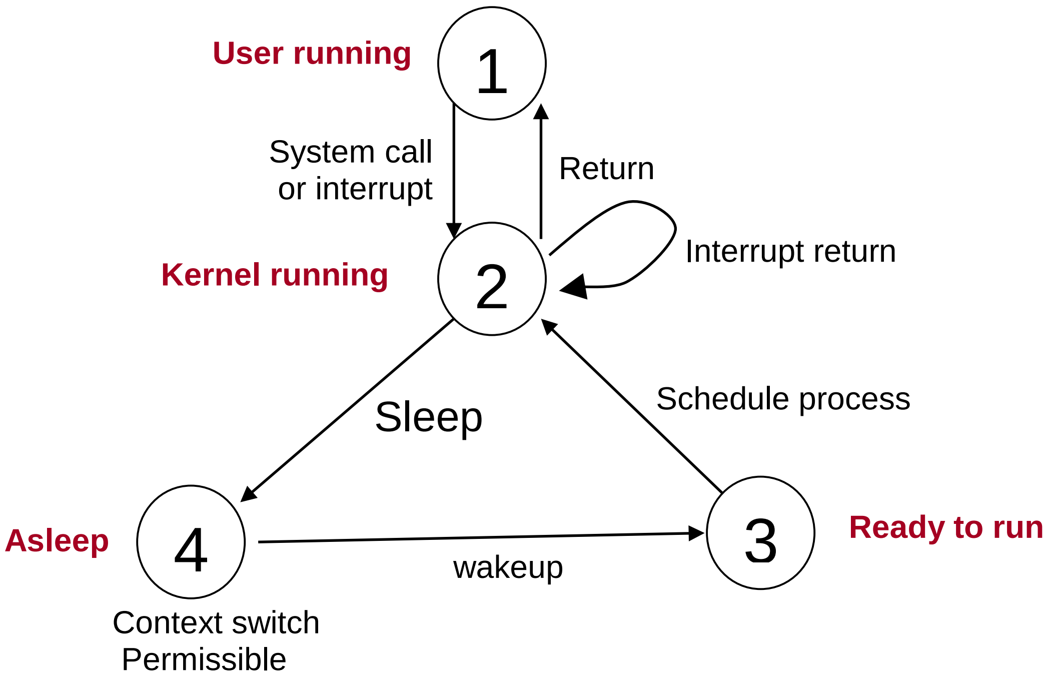


Figure 3.1.2: UNIX process states

1. User running: The process is running in user mode
2. Kernel running: The process is running in kernel mode
3. Ready to run: The process is not executing, but it is ready to run as soon as the scheduler chooses it.
4. Asleep: The process is sleeping such as waiting for I/O to complete

The UNIX process runs in two modes:

a. **User mode**

   -Can access its own instructions and data, but not kernel instruction and data

b. **Kernel mode**

   -Can access kernel and user instructions and data

When a process executes a system call, the execution mode of the process changes from user mode to kernel mode.

Self-assessment Questions

1) Kernel data structure used to define state of a process are:

   a) Process and inode table b) Inode table

   c) Region table d) Process table and uarea

2) Process table entry helps the kernel to locate the process and its ____________

   a) Inode b) Uarea

   c) Process table d) All of these

3) Which command is used to see the currently running processes on your system?

   a) cd b) pwd

   c) ps d) lse

4) What is the ready state of a process?

   a) When process is scheduled to run after some execution

   b) When process is unable to run until some task has been completed

   c) When process is using the CPU

   d) None of the mentioned

3.1.2 Layout of System memory

Each process has physical characteristics, which are managed by its virtual address space. This section discusses about model of memory management.

For understanding purpose, assume that physical memory addresses start from 0. A UNIX process has three logical sections: text, data and memory stack. Text sections contain program instructions the system executes. This section contains text addresses used for branching, data addresses used for global data variables and stack addresses used to access variables in local subroutine.

Now the problem is that machine generated addresses can't be treated as physical addresses, as there could be a possibility that addresses generated for two processes can overlap and hence, they can't execute concurrently. As the system memory is finite, it is difficult for compiler to generate addresses that do not overlap. Therefore, the compiler generates virtual addresses. The memory management subsystem of machine translates these virtual addresses into physical addresses in memory.

Kernel divides the virtual addresses into logical regions. Regions are the continuous address spaces that can be shared. The text, data and stack form separate regions for a process. Several processes can share a region. Every process owns a region table (i.e. pregion) that has information about the physical addresses. Pregion entries are maintained in process table.

Memory management architecture based on pages divides the physical memory into equal sized pages. Every memory location can be addressed by a page number and page offset.

Kernel assigns physical pages to a region. This method of physical memory allocation is more flexible than assignment of disk blocks.

Now, we can summarize as following:

Compiler generates virtual addresses for a process and stores in region table. The region table entry contains the pointer to a page table that contain the physical page numbers.

The following figure 3.1.2 shows the mapping of a process to physical memory:

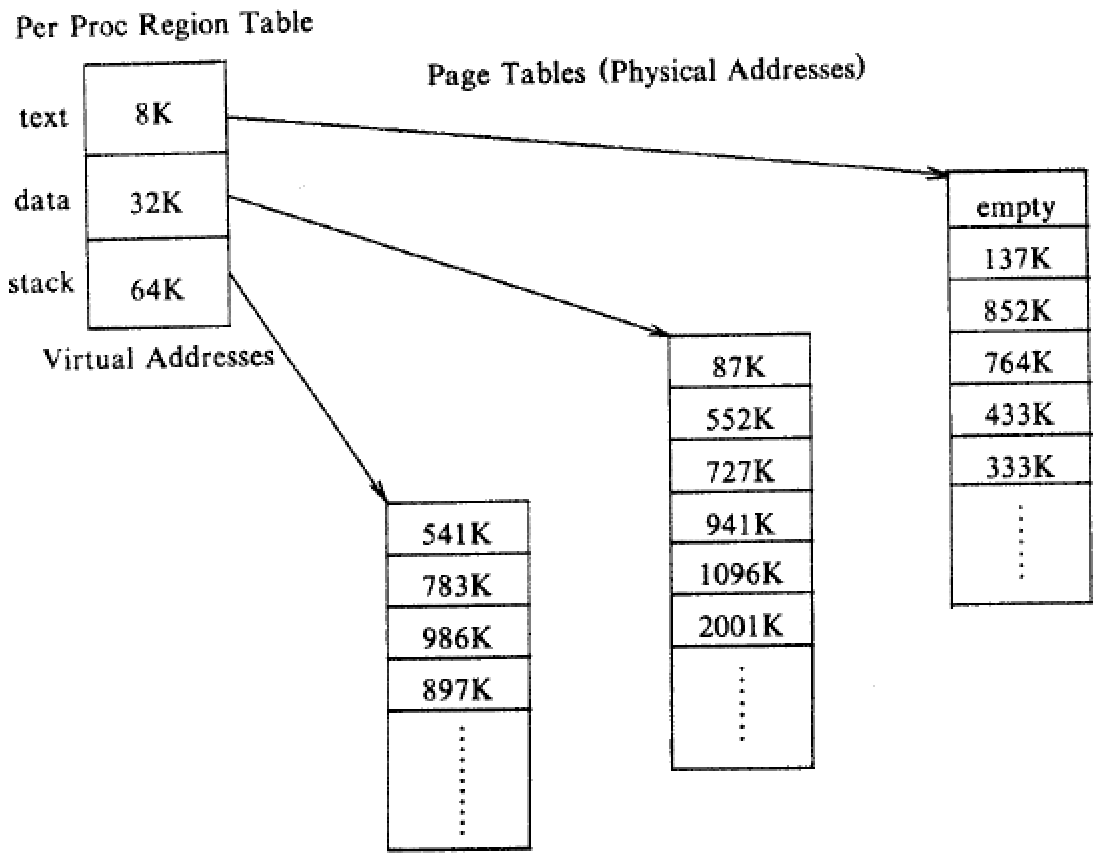


Figure: 3.1.2: Mapping virtual addresses to physical addresses.

3.1.3 The context of a process

In a multiuser, multitasking and time sharing environment, we need to store and restore status of a process, so that execution of a process is resumed from same point later. This process is also known as context switching.

The context of a process is its state:

- Text, data(variable), register
- Process region table, U Area,
- User stack and kernel stack

When a process is under execution, the system is said to be executing in the context of the process. When the kernel decides that it should execute another process, it does a context switch, so that the system executes in the context of the other process

When doing a context switch, the kernel saves enough information so that it can later switch back to the first process and resume its execution.

Context of a process can be saved at three levels:

1. user-level
2. register-level
3. system-level

1. User-level context

User level context consists of process text (program instructions), data, user stack and any shared memory which are stored in virtual address space.

2. Register context

When a process is not executing, its status is saved in register context area. Register level context consist of program counter (i.e. next instruction to run), processor status register, pointer to user or kernel stacks and general registers. Program counter tells you the next command to be executed which can be in user or kernel space. Stack pointers point to user or kernel stack.

3. System-level context:

System level context consist of Ptable entry, uarea, Per Process region (Pregion) entries, region tables, page tables.

Self-assessment Questions

5) Which of the statement is true?
   a) The machine generated addresses generated by machine can’t be treated as physical addresses
   b) The machine generated addresses generated by machine can be treated as physical addresses
   c) That machine generated addresses generated by machine are physical addresses
   d) All of these

6) Which of the following statement is true?

a) Memory management architecture based on pages divides the physical memory into unequal sized pages.

b) Memory management architecture based on pages divides the physical memory into equal sized pages.

c) Memory management architecture based on disk blocks divides the physical memory into equal sized pages.

d) Memory management architecture based on pages divides the physical memory into disk blocks.

7) In operating system, each process has its own.

a) Address space and global variables   b) Open files

c) Pending alarms, signals and signal handlers   d) All of the mentioned

8) In Linux address space defines virtual address space assigned to.

a) Processor   b) Process

c) Memory   d) Virtual Memory

9) When a process is not executing, its status is saved in __________ context area.

a) Register   b) User

c) System   d) None of these

10) The address of the next instruction to be executed by the current process is provided by the

a) CPU registers   b) Program counter

c) Process stack   d) Pipe

Summary

- UNIX is a multiuser, multitasking and timesharing system. In UNIX system many programs can run at the same time i.e. many processes can be present at the same time but each process takes its turn for running.
- Two kernel data structures are used to define state of a process are Process Table and Uarea.
- Process table entry helps the kernel to locate the process and its uarea.
- Uarea has a pointer to process table.
- Using ps command, you can see the currently running processes on your system.
- Each process has a unique id PID (i.e. process ID).
- When a process executes a system call, the execution mode of the process changes from user mode to kernel mode.
- Each process has physical characteristics which are managed by its virtual address space.
- A UNIX process has three logical sections: text, data and memory stack.
- Kernel divides the virtual addresses into logical regions. Regions are the continuous address spaces that can be shared. The text, data and stack form separate regions for a process.
- Compiler generates virtual addresses for a process and stores in region table. The region table entry contains the pointer to a page table that contain the physical page numbers.
- In a multiuser, multitasking and time sharing environment, we need to store and restore status of a process, so that execution of a process is resumed from same point later. This process is also known as context switching.
- When the kernel decides that it should execute another process, it does a context switch, so that the system executes in the context of the other process.
- Context of a process can be saved at three levels: user-level, register-level and system-level.
- When a process is not executing, its status is saved in register context area.

Terminal Questions

1. Discuss process state model for the UNIX system and set of state transitions.
2. Describe the principles of memory management for processes and kernel.
3. Explain how the operating system and hardware cooperate to do virtual memory translation.
4. Explain the components of the context of a process.

Answer Keys

Self-assessment Questions	
Question No.	Answer
1	d
2	b
3	c
4	a
5	a
6	b
7	d
8	b
9	a
10	b

Activity

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

Description:

1. Prepare a presentation (15 slides) on Process state transition.

Bibliography

e-References

- This website was referred on 3rd May 2016 while developing content for Structure of Process topic http://www.cs.kent.edu/~farrell/osf03/oldnotes/L06.pdf
- This website was referred on 3rd May 2016 while developing content for Structure of Process topic http://duartes.org/gustavo/blog/post/how-the-kernel-manages-your-memory/
- This website was referred on 3rd May 2016 while developing content for Structure of Process topic http://www.cs.umsl.edu/~sanjiv/classes/cs4760/lectures/memory.pdf

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: