Process

Process in the Unix

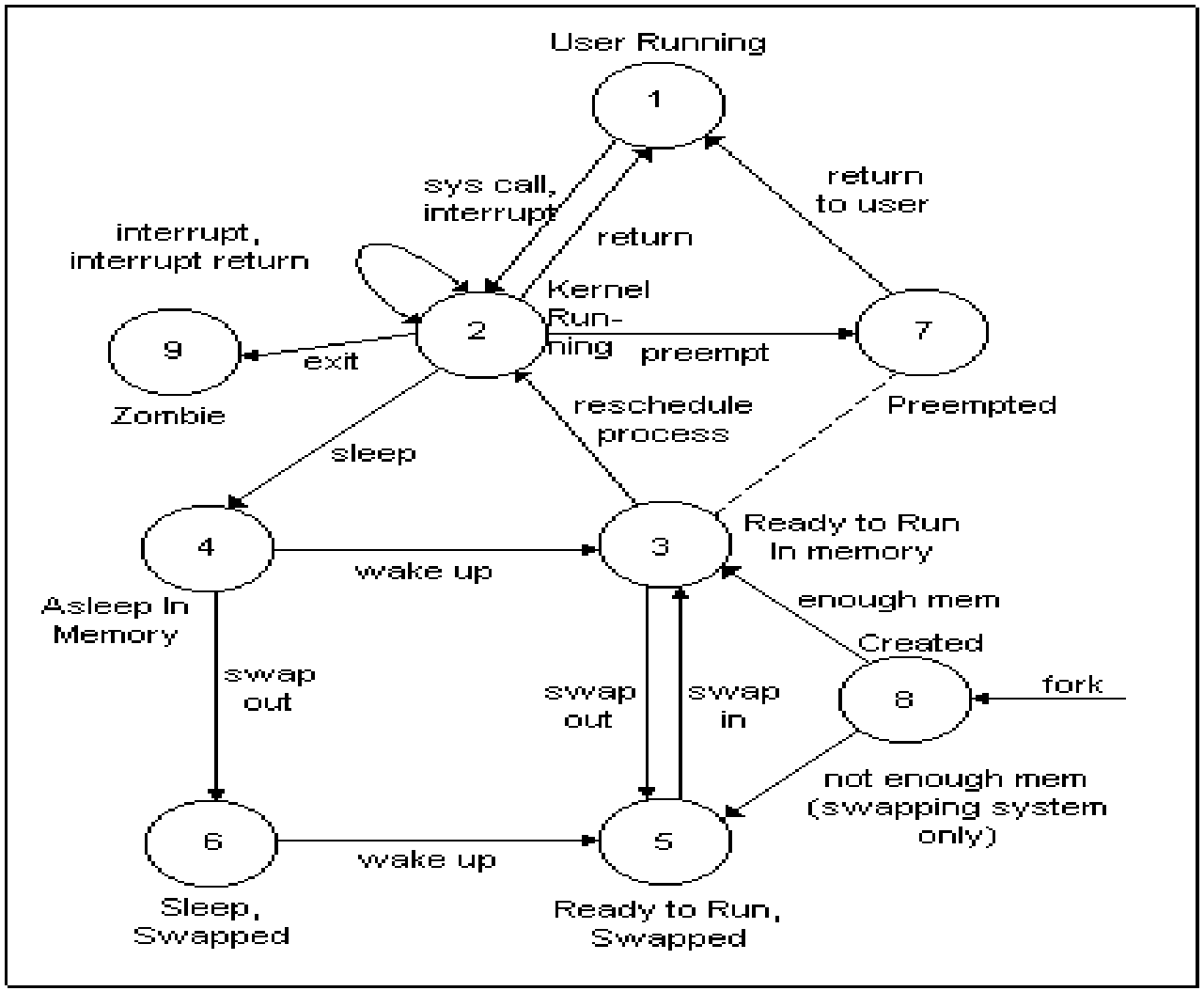
- Any program that is copied in the RAM is called as process.
- Time to time kernel creates no. of processes and schedules them and finally terminates them.
- Since Unix is multitasking OS more than one process can run at a time. Each process is uniquely identified by a number called as PID which is allotted by kernel and is in between 0 and 32767.
- One CPU runs one process at a time, kernel is responsible to manage them. It allocates time priority, swapping and so on.

The lifetime of a process

- The process is executing in user mode.
- The process is executing in kernel mode.
- The process is not executing but is ready to run as soon as the kernel schedules it.
- The process is sleeping and resides in main memory.
- The process is ready to run, but the swapper (process 0) must swap the process into main memory before the kernel can schedule it to execute.
- The process is sleeping, and the swapper has swapped the process to secondary storage to make room for other processes in main memory.

- The process is returning from the kernel to user mode, but the kernel preempts it and does a context switch to schedule another process. The distinction between this state and state 3 ("ready to run") will be brought out shortly.
- The process is newly created and is in a transition state; the process exists, but it is not ready to run, nor is it sleeping. This state is the start state for all processes except process 0.

- **The process executed the exit system call and is in the zombie state. The process no longer exists, but it layer a record containing an exit code and some timing statistics for its parent process to collect. The zombie state is the final state of a process.**



Process State
Transition

Process Table

- For various processes UNIX kernel maintains a table of fixed size called Process table
- It contains information on each currently active process
- If the process table is full, no other process can start and the error message "No more processes" is displayed if one more process is given
- If any entry in this table is removed then any new process can start

- **The status of the process can be seen by command *ps* which gives information about all the process running**
- **The basic function of Process table is to enable the scheduler (kernel) to manage processes effectively.**

init

- There are different init processes which defines the opertional level of UNIX system
- According to the level specified, the kernel initializes the required hardware
- These different operational levels helps in maintaining the system or to shut down the system and to do other important functions

Different init process

- init 0 - SHUTDOWN
- init 1 - SYSTEM MAINTANCE
- init 2 - MULTIUSER
- init 3 - REMOTE FILE SHARING STATE
- init 4 - ALTERNATIVE MULTIUSER
- init 5 - STOP UNIX & GO TO FIRMWARE
- init 6 - REBOOT

Note: init 3,4 and 5 are normally used by O.S. developers.

Multiuser Mode (init 2)

- To go to multiuser mode, type init 2
- This will initiate a series of events which ends with the gettys being started for each tty, thus allowing other users to log in
- The /etc/inittab gives definitions for how to start gettys for different ttys under level 2
- At the least, two shell programs /etc/bcheckrc, and /etc/rc are given by default. These are executed before the gettys are started

- /etc/bcheckrc prompts for the correct time and date and it asks whether to check the file systems. If you say ("y"), it will run the fsck command.
- /etc/rc it starts up the processes (except for gettys) that required in multiuser mode, such as the cron, printers controllers, and etc, it also mounts the file systems and may also start system accounting, error logging, and system activity logging.

- Getty
- Monitors the terminal lines configured on a system by the init 2 process
- It reads data pertaining to the speed & terminal settings for a line from /etc/gettydefs file.
- Whenever the terminal is on & connected to the host it displays the login prompt on the screen & waits for the user to enter login user name & then for the valid password .

/etc/gettydefs

- This file contains information used by getty to set up the speed & terminal settings for a line.
- It supplies information on how the login prompt should look like .
- It also supplies the speed information to be tried next if the user indicates that the current speed is not correct by typing a BREAK character .s

Processes status

ps :- Process status

- **Lists all the active processes that has been started by the user**

  ***syntax* : $ ps [options]**

  options

-t ttyno. To list the processes started by the specified terminal (ttyno.)

eg. **: $ ps -t 005**

-u usernameTo list all the processes started by the specified user (username)

eg. **: $ ps -u rakesh**

-f To list more information about the process

eg. : $ ps

PID	TTY	TIME	COMMAND
123	003	0:01	sh
124	003	0:00	ps

PID - Process ID

TTY - Terminal controlling the process

TIME - Execution time of the process

COMAND - Command name

Killing Process

Kill :- To terminate a process

- **To terminate a background process or process running on different terminal kill command can be used**
- **To terminate a process its process ID should be known.**
- **User can terminate his own processes, but root can terminate any process running on the system**

syntax **: $ kill -<signal number> <process ID>**

- **By default if no signal number is specified then default is 15 which can be ignored by the process and terminates the process when the system is free**
- **To terminate the process immediately specify signal number as -9**
- ***eg.*** **: kill -9 124**
- **Termination of the process is notified by the following message**
- **124 : Terminated**

Running process in background

- To place a command in the background put an ampersand (&) at the end of the command line
- After the command is placed in the background system displays the job number and returns to the shell prompt
- When the job is finished then the message is send to the user.
- Background jobs are killed if user logs out

eg. : $ banner CMS > sample & then system returns the following message [1] 1446

Timing Processes

- The time command :- when faced with several versions of a program , it becomes necessary to find out the drainage which the command or program makes on the system resources . The time command does this work . It accepts the command to be timed as its argument & then not only executes the command to be timed as its argument & then not only executes the program but displays the time usage in the terminal as well.