

# UNIX

## Processes and Related Commands

# Lesson Objectives

## ➤ **UNIX processes:**

- Parent and Child processes
- Process Status Command – ps
- Running processes in background mode
- Terminate process
- Process scheduling



# What is a Process?

## ➤ **Characteristics of processes:**

- Process is an instance of program in execution.
- Many processes can run at the same time.
- Processes are identified by the Process Identifier.
- PID is allocated by kernel.

# Concepts

- On logging to a system, a process is set up due to execution of shell.
- Shell is the parent process for every other process setup due to the execution of commands.
- Every process, with the exception of PID 0 processes, has a parent process.
- Parent process waits for death of child process before resuming execution.

# Running a Command

- **ls command: Steps for running a Unix command**
- The shell performs a fork. This creates a new process that the shell uses to run the ls program.
  - The shell performs an exec of the ls program. This replaces the shell program and data with the program and data for ls and then starts running that new program.
  - The ls program is loaded into the new process context, replacing the text and data of the shell.
  - The ls program performs its task, listing the contents of the current directory .

# PS Command

➤ **ps command displays characteristics of a process.**

➤ **Syntax:**

```
ps [ option [ arguments ] ... ]
```

➤ **Options:**

- -f - full form
- -u- details of only users processes
- -a- all processes details
- -l - detailed listing
- -e- system processes

**ps**

```
$ ps
  PID   TTY      TIME CMD
   599   tty0    00:00:00 sh
   613   tty0    00:00:00 ps
$ _
```

# Example

## ➤ Output of ps -l command:

```
$ ps -l
  F S      UID      PID  PPID  C PRI  NI           ADDR      SZ        WCHAN      TTY          TIME C
MD
20 R      201      599    598   3  47  24 fb11c8b0    60             -   ttty0      00:00:00 s
h
20 0      201      625    599   1  48  24 fb11ca08   164             -   ttty0      00:00:00 p
$
$ _
```

# Process in Background Mode

## ➤ Processes can run in foreground or background mode.

- Only one process can run in foreground mode but multiple processes can run in background mode.
- The processes, which do not require user intervention can run in background mode, e.g. sort, find.
- To run a process in background, use & operator

- `$sort -o emp.lst emp.lst &`

## ➤ nohup (no hangup) - permits execution of process even if user has logged off.

- `$nohup sort emp.lst &` (sends output to nohup.out)



# Kill Command

➤ **Kill Command- Used to terminate a process**

➤ **Syntax :**

- `kill [ -signumber ] pid ...`

➤ **Example:**

- `$kill 1005` (default signal 15) - kills job with pid 1005
- `$kill -9 1005` - sure killing of job
- `$kill 0` - kills all background process

# Details

## ➤ Scheduling Policy:

- *time-sharing* technique
- Several processes are allowed to run "concurrently," which means that the CPU time is roughly divided into "slices," one for each runnable process.
- The scheduling policy is also based on process priority
- In UNIX, process priority is dynamic.

## Continued...

- **Processes are traditionally classified as "I/O-bound" or "CPU-bound."**
  - **I/O-bound Processes:**  
Make heavy use of I/O devices and spend much time waiting for I/O operations to complete.
  - **CPU-bound Processes:**  
Are number-crunching applications that require a lot of CPU time.

# Continued...

## ➤ **Processes can also be classified as:**

- **Interactive processes:**

These interact constantly with their users, and therefore spend a lot of time waiting for key presses and mouse operations.

- **Batch processes:**

These do not need user interaction, and hence they often run in the background.

- **Real-time processes:**

- Should never be blocked by lower-priority processes.
- Should have a short response time.

# nice and wait command

➤ **nice - runs a program with modified scheduling priority.**

➤ **Syntax :**

```
nice [OPTION] [COMMAND [ARG]...]
```

— \$ nice cat chap?? | nice wc -l > wclist &

➤ **Wait - waits for child process to complete.**

➤ **Syntax :**

```
wait [ process id... ]
```

— \$wait 138 - waits for background job with pid 138

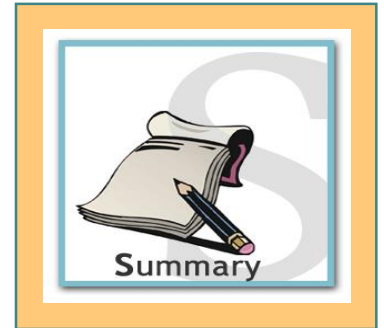
# cron

- A system daemon which performs a specific task at regular intervals
- The command and schedule information is kept in the directory `/var/spool/cron/crontabs` or in `/usr/spool/cron/crontabs`.
- Each user has a crontab file. cron wakes up periodically and executes any job that are scheduled for that minute.
- Only users who are listed in `/etc/cron.allow` or not listed in `cron.deny` can make an entry in the crontab.
- **Crontab <filename>** -used to make an entry in the crontab file.
  - where the file contains the commands to execute

MIN	HOUR	DOM	MOY	DOW	COMMAND
(0-50)	(0-23)	(1-31)	(1-12)	(0-6)	---
\$ 0	18	*	*	*	/home/gather

# Summary

- **Unix processes**
- **Process related commands**
  - ps
  - nohup
  - wait
  - kill
  - nice
- **Background processes**



# Review Questions

## ➤ Complete The Following :

- A unique number called the \_\_\_\_\_ identifies each process.
- Processes using heavy i/o are called as \_\_\_\_\_

## ➤ True / False

- A signal number of 9 is used, by default, by the kill command to terminate a process.
- You can kill any process, including the system process, using the kill command.

