Operating System Lab

Lab 1
Basic Process Concepts

Write a Program in C

- Write a simple C program which use infinite loop.
- Use “gedit” as Editor.
- Use “gcc” as compiler.
- Do not use “conio.h”.
- No need to use “clrscr()” and “getch()”.
- Compiler creates an executable file named “a.out”.
- Run your program as “./a.out”.
- Press “ctrl+c” to stop the program.

ls Command

- “ls” -> List files in current directory.
- “ls -l” -> Full listing about all files in current directory.
- “ls -l abc.txt” -> Full listing about file abc.txt.
- “ls” is command, “-l” is option, “abc.txt” is argument.
- Options are special type of arguments.
- Options are fixed in numbers.
- Options handles basic behaviour of commands.
- Arguments are variable in number.
- Commands have to operate on given arguments.

ps Command

- “ps” -> Show processes running on current terminal.
- “ps -f” -> Full listing of processes running on current terminal.
- Notice Parent-Child relationship among processes.
- “ps -ef” -> All processes running in the system.
- “tty Command” -> Name of terminal you are working on.
- “ps -t /dev/pts/0” -> Processes running on terminal /dev/pts/0
- Open another terminal. Run your C program their.
- Go to another terminal and run “ps -ft /dev/pts/x” Command.
- See PID and PPID of your C program.

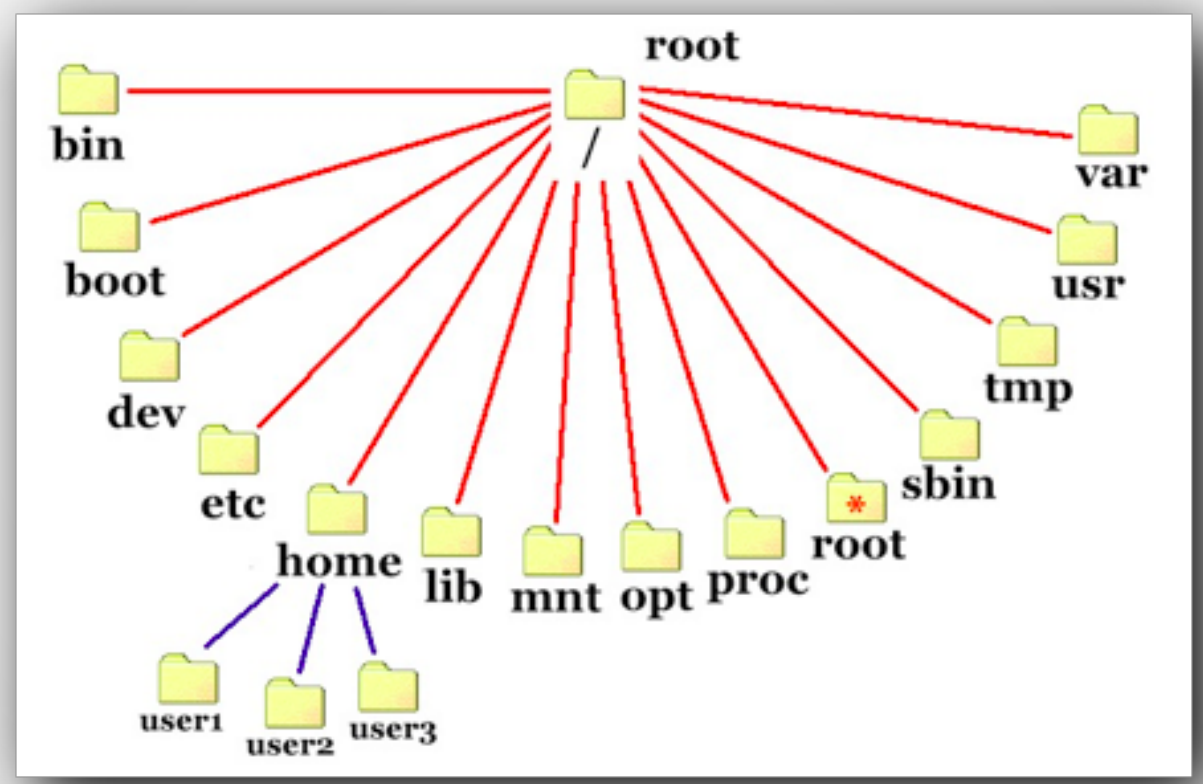
What is Process?

Program in Execution is Process.

Some More Commands

- “gedit” -> You already seen.
- “firefox” -> Open firefox web browser.

File System Hierarchy



Where are Commands

- All commands are stored in either of these directories:

  /home/hduser/bin, /usr/local/sbin, /usr/local/bin, /usr/sbin, /usr/bin, /sbin, /bin, /usr/games, /usr/local/games,

- Linux searches only these directories for commands. If your command is stored someware else you have to give full pathname.
- These commands are executables like a.out.
- If you copy a.out to one of theses directories then you can directly use a.out (not ./a.out).
- “whereis ls” -> Where is your ls command.

Standard streams

- There are three standard streams related to each process:

  1). stdin (Standard Input) -> Related to keyboard

  2). stdout (Standard Output) -> Related to Monitor

  3). stderr (Standard Error) -> Related to Monitor

Find PID and PPID of C Program

- getpid() -> Returns processID of program.
- getppid() -> Return processID of parent.
- pid_t getpid();
- pid_t getppid();

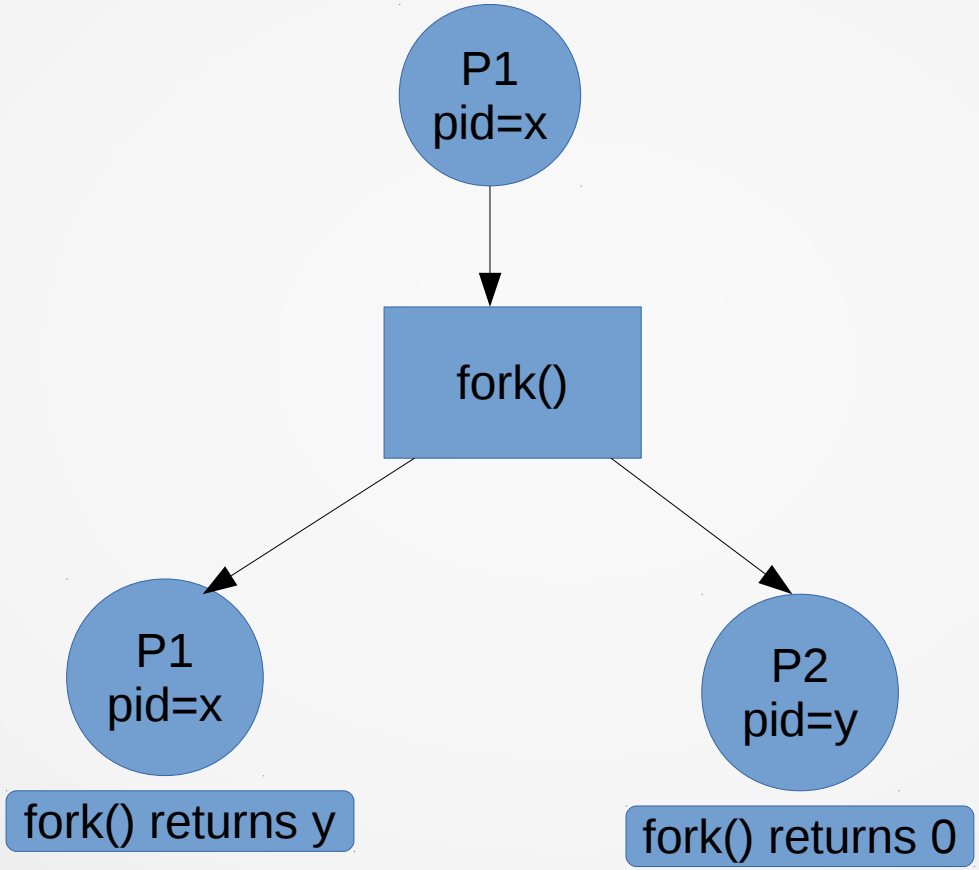
What is pid_t

- pid_t has been typedef into int in header-file “unistd.h” and “sys/types.h”.
- You can directly use Integer.
- Using pid_t makes type system independent.

fork() System Call

- Creates new child of given process. After that both starts seperate executions.
- fork() copy everything of parent into child. Except PID and PPID.
- Executable code and standard streams are also will be copied as it is.
- fork() returns pid_t type.
- fork() returns 0 in child process and PID of child in parent process.

fork() System Call



fork() System Call

- Flush out all streams before fork.

execlp() System Call

- Parent and child both are repeating same code.
- Creating new process is useful only if we are doing something new by parent or child.
- execlp() replaces the calling process image with a new process image. This is called overlaying.

Process Death

- Kernel manages a table which contains entries for each process.
- Before death a process writes its exit status on this table.
- Parent of this process enquire its child's exit status. Till the time the child remains as zombie.
- Zombie is a dead process whose entry still exists in process table.
- After enquiry by parent process table entry gets deleted.
- If parent is dead before child then the child is called orphan. And after child's death a special process init takes this responsibility.

wait() System Call

- Orphans should not allowed.
- We can make parent waiting for child's death using wait() system call.

Task 1

Apply all discussed program.

Task 2

Write a program in C to create a new process. Parent process will open gedit and child will open firefox.

Task

- Write a C program using fork() to create copy of the process.
- Write a C program using fork() to create copy of the process and then modify a variable in both the processes separately.
- Write a C program using fork(), getpid(), and getppid() to print PID of parent and child process along with their parents PIDs.
- Write a C program using fork(), wait(), and execlp() and use child process for running other program (e.g. ls, echo, date commands).