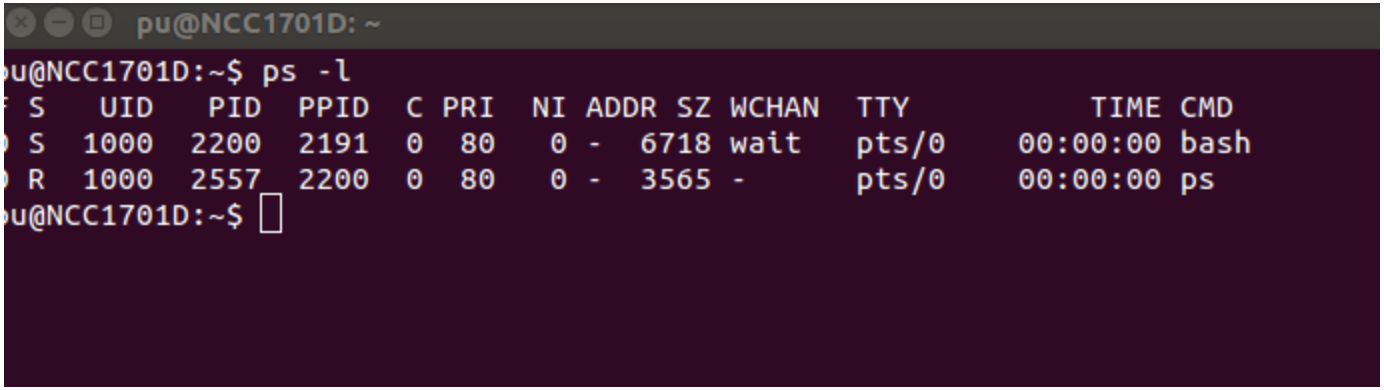
Linux Process Control

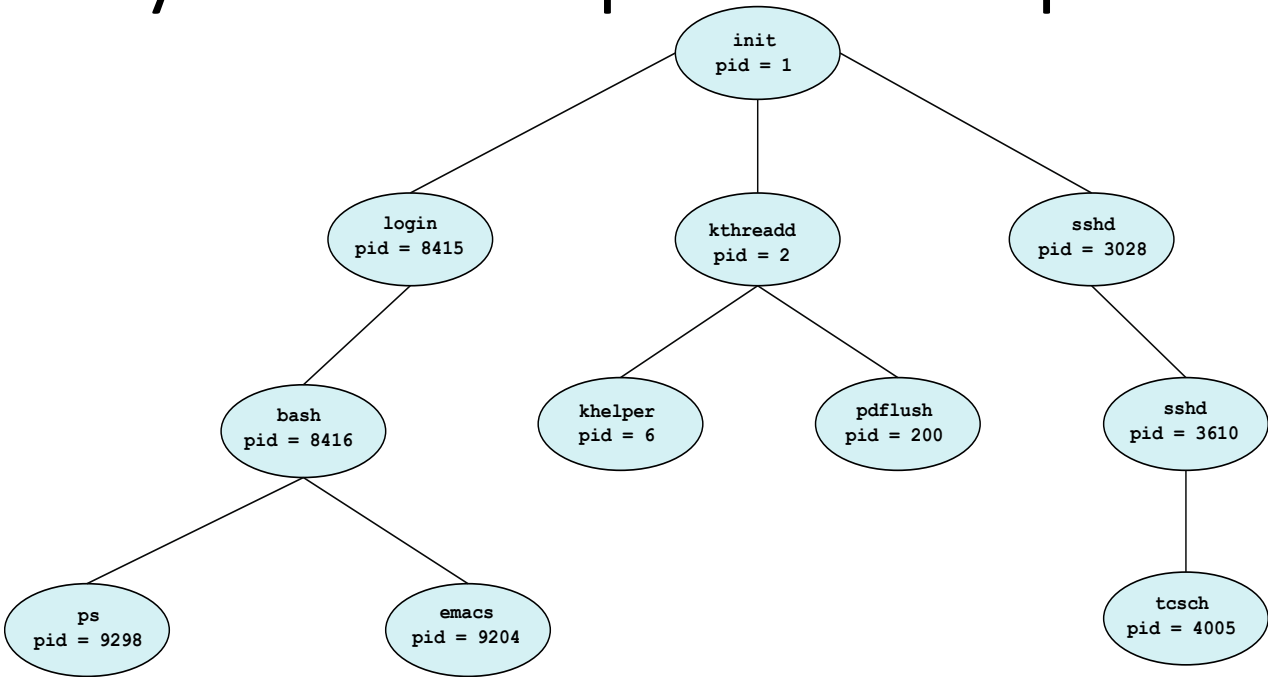
Process

- Process is a running program in memory
- Process is identified by process id or “pid” assigned to it on creation by Linux system
- Process is created by other process called parent
  - So, each process will have a parent process id except init process (process 0, the first os process)



Process Tree

- Process in Linux is created by parent so process will have a tree relation ship
- When boot up process 0 called init is created by the system as the parent of all processes

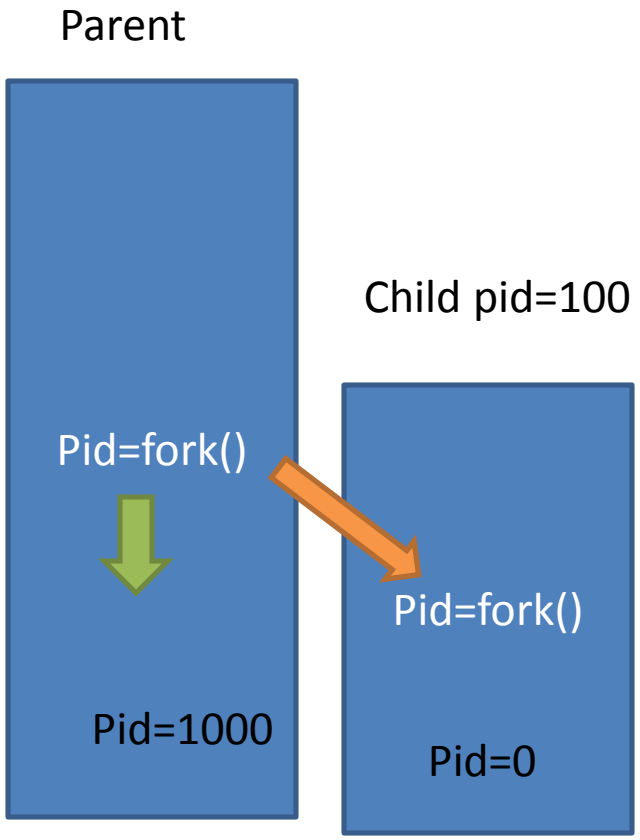


Process creation

- Linux use a set of system call to perform process creation
  - Fork – create new process
  - Wait – parent call wait to wait for child to finish
  - Exit – process call this function to exit from execution
  - Exec – process use this to execute command

fork

- Process call this function to create a new process
  - Pid = fork()
- When this function is called, a new process is created while parent still running
- If pid return 1 this process is a child process
- If pid return a number it is child process id
- If pid < 1 it is an error



Child process

- Linux OS will copy everything from parent to create a child process
- So child process share many thing from parent
  - Memory
  - File descriptor
  - Open socket

Wait and exit

- Parent process call wait function to wait for child process
  - Pid_t wait(int * status)
  - Int status;
  - Wait(&status);
- Status is a return value when child process exit using exit function
  - Int exit(exit_value);

exec

- Exec is a set of system calls used to execute linux program
- Program being exec will be loaded into memory and replace the process image of current process that call exec
- One of the very easy on is execlp
  - execlp(command,arg0,arg1,…,NULL);
  - First argument is executable command, next is arg0,arg1,arg2… end with NULL

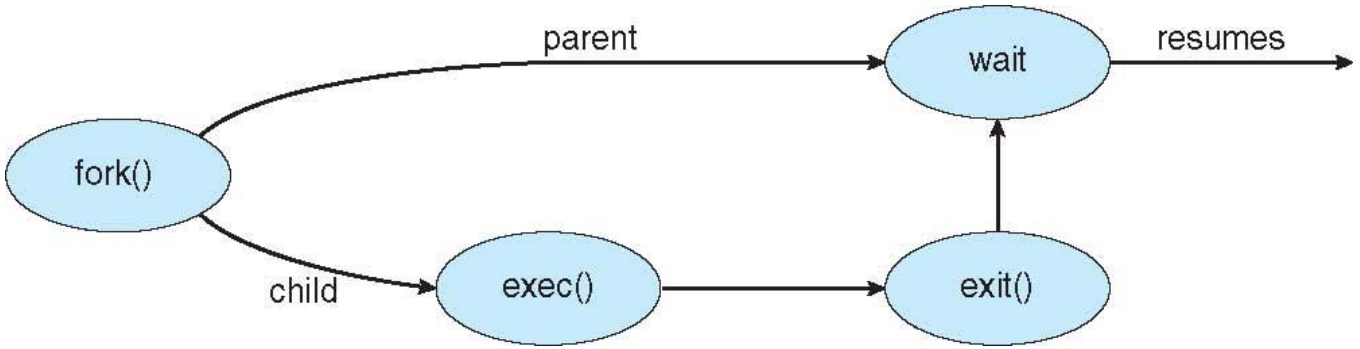
example

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
   pid_t pid;
   /* fork exec example */
   pid = fork();

   if (pid<0) { /* error */
               fprintf(stderr,"fork failed\n");
               return 1;
    } else if (pid == 0) { /* chjild process */
               execlp("/bin/ls","ls",NULL);
    } else { /* parent */
       wait(NULL);
       printf("Child complete\n");
    }
    return 0;
}
```

How it works



System

- System start shell and run command for you from c program

```
#include <stdio.h>

int main()
{
        printf("This is parent\n");
        system("ps -l");
}
```

SIGNAL

- Signal is simple mechanism used by linux to notify process to take some action
- Signal is a kind of software interrupt
- Example
  - Pressing control-C will generate SIGINT to process that is running in the backgroud causing that process to terminate
  - Try
    - Sleep 1000
    - Press ctrl-c
      - This will send signal SIGINT to process and make it terminate

KILL

- kill is used to send signal to process
  - Int kill (pid_t pid,int signal)
- This signal is used to control some operation of process
  - SIGKILL – kill process
  - SIGQUIT – tell process to terminate
- Process that exchange signal must belong to the same user

SIGNAL Handling

- When process get a signal there are a few choices
  - Let the default action happen mostly cause a process to terminate
  - Tell the system to ignore it
  - catch the signal and install your signal handling (except SIGKILL which can not catch)

Example: catching ctrl-c

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>

/* Signal Handler */
void handler(int signo)
{
        if (signo == SIGINT)
                printf("ctrl-c pressed\n");

}
int main()
{

        if ( signal(SIGINT,handler) == SIG_ERR) {
           printf("Error\n");
           return 0;
        }
        while (1)
          sleep(1);
        return 0;
}
```

alarm

- Alarm generate SIGALRM alarm every period of time specified to
  - alarm(int timer)
- This is usedfull function since you can put process suspended using
  - Int pause(void)

reference

- SIGNAL
  - http://www.thegeekstuff.com/2012/03/linux-signals-fundamentals/

exercise

- Write program to trap SIGALARM and display time every 10 second

Homework

- Writing a simple shell program
  - Display command prompt
  - Read input string
  - Execute command typed
  - Loop back to get the string again until comand “exit” is typed