Process

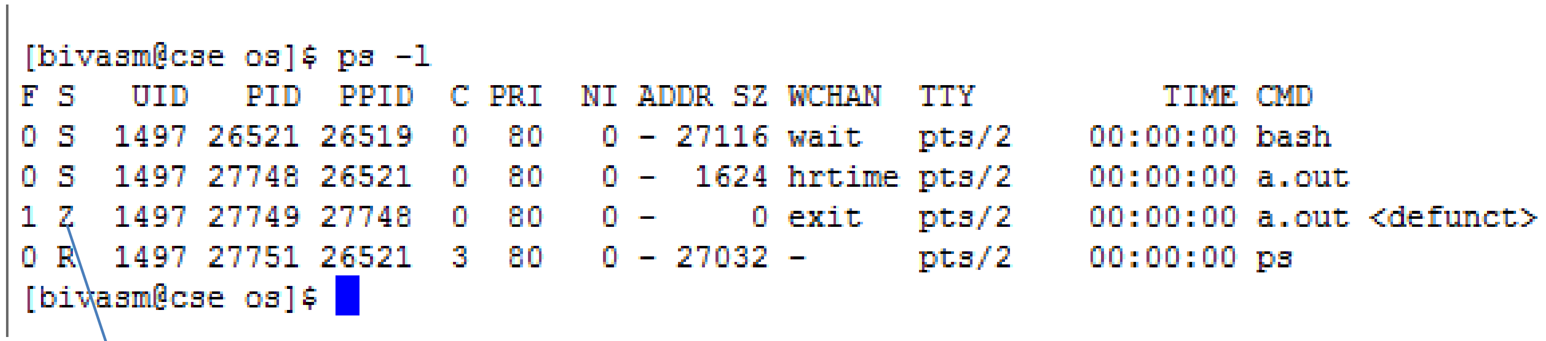
fork

```
Main()
{
Pid=fork();
If(pid==0)
{
        Child process
        getpid();
        sleep(20);
}
else
{
        Parent process
        getpid();
        sleep();
}
```

Orphan process

```
main()
{
pid=fork();
If(pid==0)
{
            getpid();
            getppid();
            sleep(10);
            getppid();
}
else
{
            getpid();
            getppid();
}
```

Process table



```
[bivasm@cse os]$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1497 26521 26519  0  80   0 - 27116 wait   pts/2    00:00:00 bash
0 S  1497 27748 26521  0  80   0 -  1624 hrtime pts/2    00:00:00 a.out
1 Z  1497 27749 27748  0  80   0 -     0 exit   pts/2    00:00:00 a.out <defunct>
0 R  1497 27751 26521  3  80   0 - 27032 -      pts/2    00:00:00 ps
[bivasm@cse os]$
```

Zombie

```
Main()
{
Pid=fork();
If(pid==0)
{
            printf(“First Child process”)

}
else
{
            dip=fork()
            if(dip==0)
            {
                        printf(“second child”)
            }
            else
            {
            cpid=wait(0);
            printf(“child died %d”, cpid);
            cpid=wait(0);
            printf(“child died %d”, cpid);
            printf(“Parent”);

}
```

```
Main()
{
Pid=fork();
If(pid==0)
{
            printf(“child process”)
            exit(i);
}
else
{           wait(&status);
            printf(“Parent process”);


}
```

Normal termination

Update	0

Abnormal
termination

0	Update

```
pid_t waitpid(pid_t pid, int *statusPtr, int options);
```

```
int main (){
    int pid;
    int status;

    printf("Parent: %d\n", getpid());

    pid = fork();
    if (pid == 0){
        printf("Child %d\n", getpid());
        sleep(2);
        exit(EXIT_SUCCESS);
    }

//Comment from here to...
    //Parent waits process pid (child)
    waitpid(pid, &status, 0);
    //Option is 0 since I check it later
```

```
Main()
{
        printf(“before”);
        execl(“usr/guest/ex2”, “ex2”, (char*)0);
        printf(“after”);
}
```

```
main(int argc, char* argv[])
{                                        ./Ex1 /usr/guest/ex2  ex2 hello world
          printf("before");
Ex1
          execl(argv[1],argv[2], argv[3], argv[4], (char*)0);
          printf("after");
}
```

Ex2

```
main(int argc, char* argv[])
{
          printf("%s %s %s", argv[0], argv[1], argv[2]);
}
```

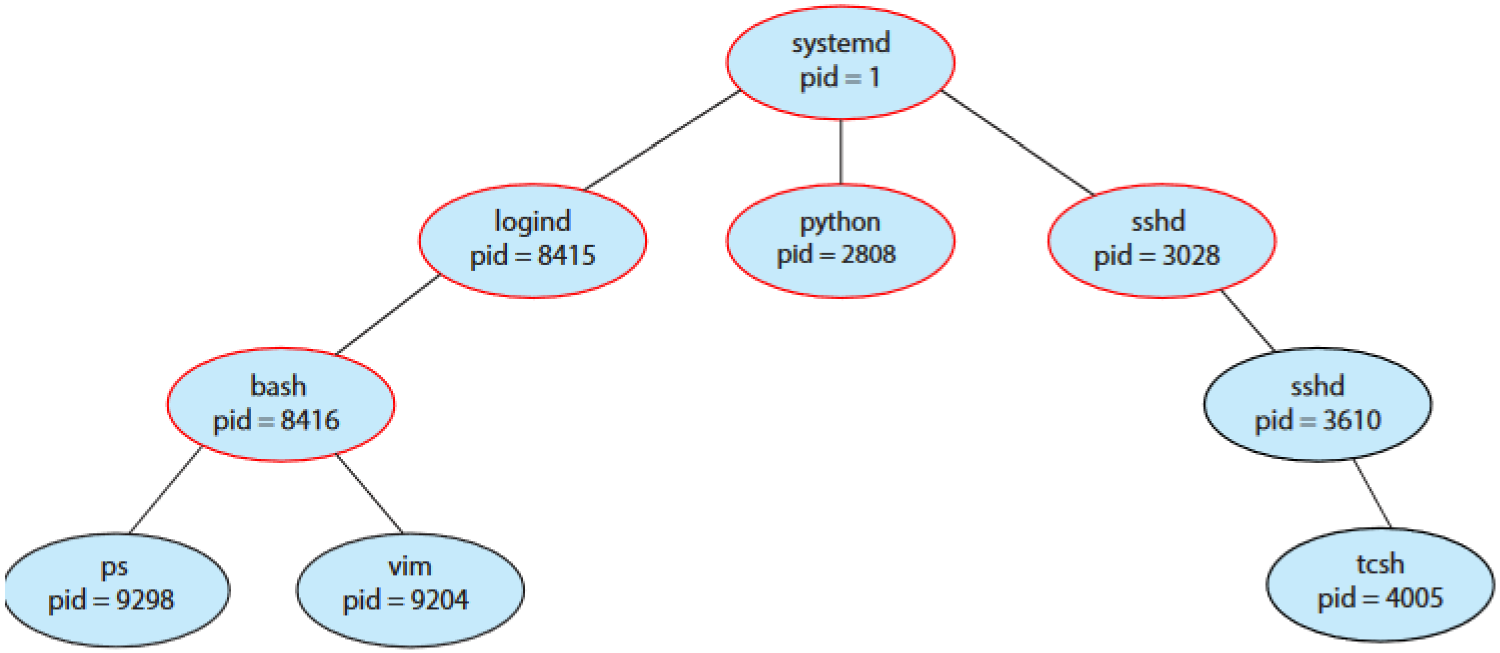
```
main(int argc, char* argv[])
{                                              ./Ex1 /bin/ls  ls -l
        printf("before");
Ex1
        execl(argv[1],argv[2], argv[3], argv[4], (char*)0);
        printf("after");
}
```

Execv(path, temp)

Execvp(file, temp)

```
Temp[0]="ex2"
Temp[1]="hello"
Temp[2]="world"
Temp[3]='\0'
Execvp(temp[0], temp)
```

```
Ex2
Printf(argv[0], argv[1], argv[2])
Ex2  hello world
```


systemd
pid = 1
logind
pid = 8415
python
pid = 2808
sshd
pid = 3028
bash
pid = 8416
sshd
pid = 3610
ps
pid = 9298
vim
pid = 9204
tcsh
pid = 4005

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

int main()
{
pid_t pid;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
       fprintf(stderr, "Fork Failed");
       return 1;
    }
    else if (pid == 0) { /* child process */
       execlp("/bin/ls","ls",NULL);
    }
    else { /* parent process */
       /* parent will wait for the child to complete */
       wait(NULL);
       printf("Child Complete");
    }

    return 0;
}
```