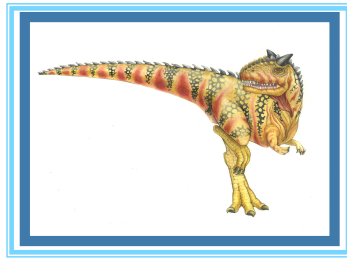


Chapter 3: Processes



Process Creation in Unix

```
int main()
{
    pid_t pid;
    /* fork another 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 */
        wait (NULL);
        printf ("Child Complete");
    }
    return 0;
}
```





Process Creation in Unix

```
int main() {
```

```
    pid = fork();
```

parent
process

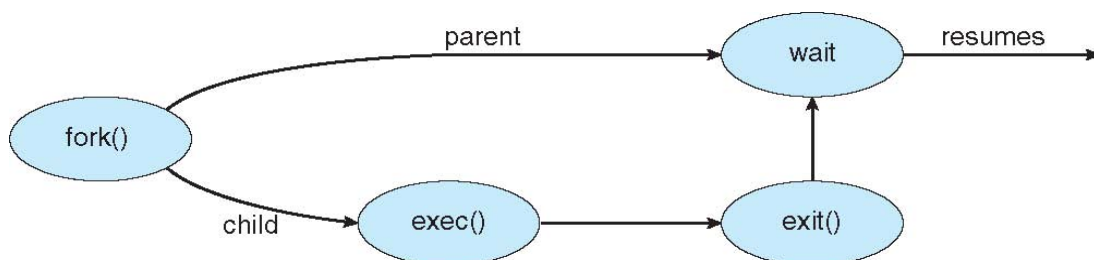
new child process
process id = 1234

```
// pid is 1234
if (pid < 0) {
    fprintf(stderr, "Fork
Failed");
    return 1;
}
else if (pid == 0) {
    execlp("/bin/ls", "ls",
NULL);
}
else {
    wait (NULL);
    printf ("Child
Complete");
}
return 0;
```

```
// pid is 0
if (pid < 0) {
    fprintf(stderr, "Fork
Failed");
    return 1;
}
else if (pid == 0) {
    execlp("/bin/ls", "ls",
NULL);
}
else {
    wait (NULL);
    printf ("Child
Complete");
}
return 0;
```



Process Creation





Process Creation in Win32

```
int main(VOID) {
    //...
    // create child process
    if (!CreateProcess(NULL, // use command line
        "C:\\WINDOWS\\system32\\mspaint.exe"
        NULL, //inherit process handle
        NULL, //don't inherit thread handle
        FALSE, //disable handle inheritance
        0, // no creation flags
        NULL, //use parent's environment block
        NULL, //use parent's existing directory
        &si, &pi)) {
        fprintf(stderr, "Create Process Failed");
        return -1;
    }
    WaitForSingleObject(pi.hProcess, INFINITE);
    printf("Child Complete");
}
```



Quiz: fork()

- What are the outputs?
(parent pid: 2600
child pid: 2603)

```
int main()
{
    pid_t pid, pid1;

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

    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        pid1 = getpid();
        printf("child: pid = %d", pid); /* A */
        printf("child: pid1 = %d", pid1); /* B */
    }
    else { /* parent process */
        pid1 = getpid();
        printf("parent: pid = %d", pid); /* C */
        printf("parent: pid1 = %d", pid1); /* D */
        wait(NULL);
    }

    return 0;
}
```





Quiz: fork()

- What is the output

```
int value = 5;

int main()
{
    pid_t pid;
    pid = fork();

    if (pid == 0) { /* child process */
        value += 15;
        return 0;
    }
    else if (pid > 0) { /* parent process */
        wait(NULL);
        printf("PARENT: value = %d",value); /* LINE A */
        return 0;
    }
}
```



Quiz: cascading fork(s)

- How many processes are created?

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

int main()
{
    /* fork a child process */
    ① fork();

    /* fork another child process */
    ② fork();

    return 0;
}
```





Quiz: fork()

```
main() {
    int a, x, y;
    a=-5;
    x=-15;
    y=-20;
    a=fork();

    if(a==0)
        y=fork();

    printf("x=%d y=%d a=%d\n", x, y,
a);
}
```

Current process ID = 100

New process ID = parent process ID +
1



Quiz: fork()

```
main() {
    int a, x, y, n;
    a=-5;
    n=1;
    x=-15;
    y=-20;
    a=fork();

    if(a==0)
        y=fork();

    while (n<3){
        if(y==0)
            x=fork();
        n++;
    }
    printf("x=%d y=%d a=%d\n", x, y,
a);
}
```

Current process ID = 100

New process ID = parent process ID +
1

