

OS – Fork & Exec

2013210111 남세현

Fork는 그 시스템콜을 요청한 프로세스의 **현재 메모리**를 새롭게 할당받은 메모리 공간에 복제하여 **새로운 프로세스**를 만드는 시스템콜입니다.

새로 만들어진 프로세스와 원래 프로세스끼리는 **모든게 다 똑같지만, 다른 부분이 하나** 있습니다. 바로 **process id, 즉 pid**입니다. Pid는 **fork() 함수의 반환값**으로,

1. 내가 **부모** 프로세스면 자식 프로세스의 pid값을,
2. 내가 **자식** 프로세스면 0
3. 에러가 발생하면 음수

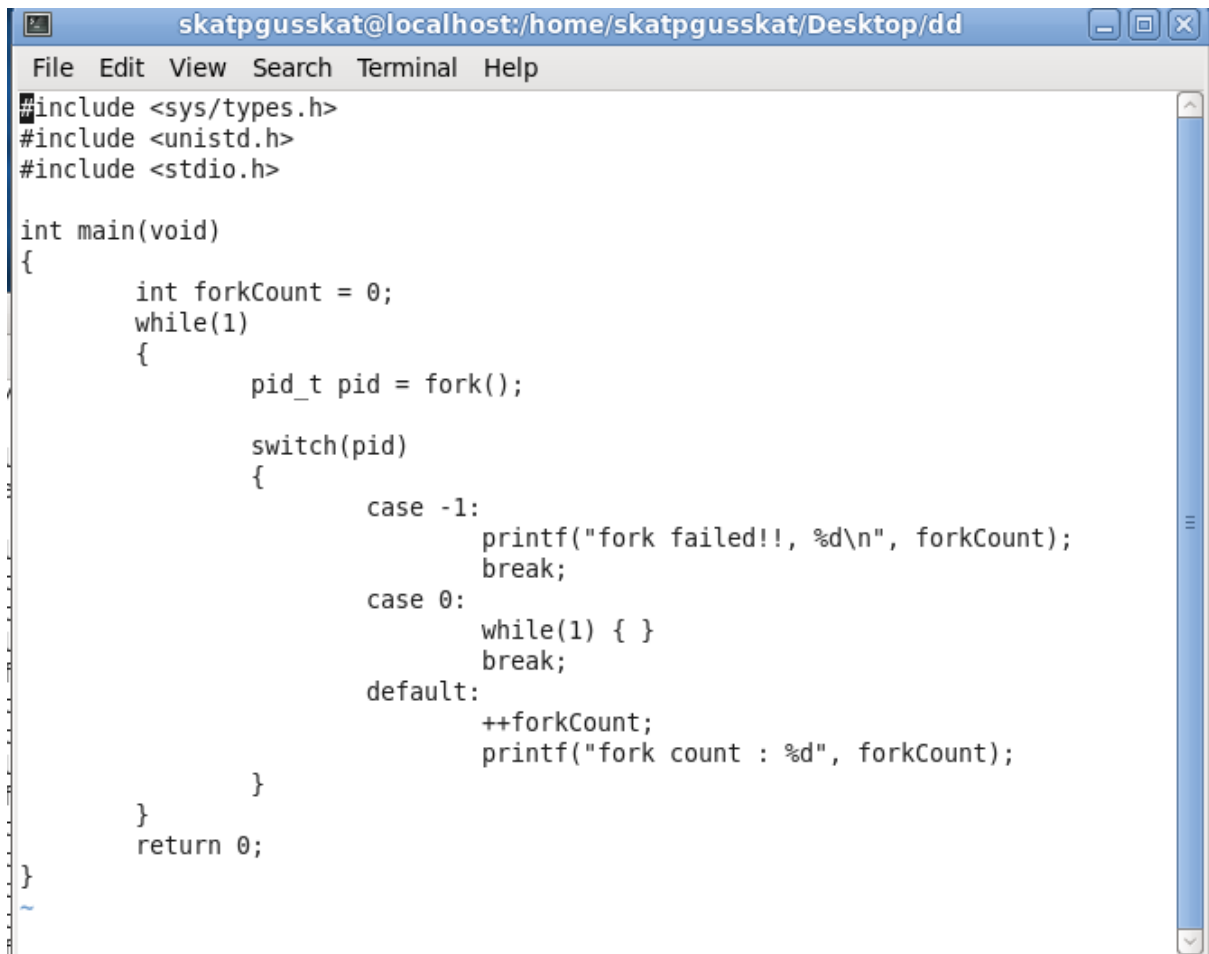
의 값을 가지게 됩니다.

Exec는 현재 프로세스의 메모리를 **싹 다 밀어버리고** 그 자리에 새로운 프로세스를 만드는 것입니다.

그렇기 때문에 원래 있던 PC(Program Count), Code Segment, 그 외에도 원래의 값을 덮어쓰므로 Exec 함수 뒤에 있던 행동은 당연히 취할 수 없습니다.

본 보고서의 다음 장에서, Fork의 실제 사용방법을 알아보면서 Fork로 만들 수 있는 프로세스의 최대 갯수를 구해보겠습니다

Fork의 최대 수

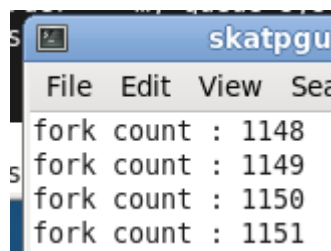


```
skatpgusskat@localhost:/home/skatpgusskat/Desktop/dd
File Edit View Search Terminal Help
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int main(void)
{
    int forkCount = 0;
    while(1)
    {
        pid_t pid = fork();

        switch(pid)
        {
            case -1:
                printf("fork failed!!, %d\n", forkCount);
                break;
            case 0:
                while(1) { }
                break;
            default:
                ++forkCount;
                printf("fork count : %d", forkCount);
        }
    }
    return 0;
}
```

위에서 알아본 것 처럼, pid의 값을 통해 부모 프로세스를 찾아낼 수 있습니다. 부모 프로세스의 경우에만 계속적으로 fork를 실행하는 루프를 수행할 수 있도록 코드를 작성하였습니다.



```
skatpgu
File Edit View Sea
fork count : 1148
fork count : 1149
fork count : 1150
fork count : 1151
```

약 3분정도 지났을 때 1000개 이상의 프로세스가 만들어졌고

그 이후로 20분 경과 후에는 시스템이 반응하지 않았습니다.

위 코드를 실행하는 PC의 환경에 따라 다르겠지만, 본 작성자의 환경(VMWare)에서는 약 2시간 후 총 3972개의 process가 만들어졌습니다.