

1.2절. 설명

PPID가 -1이 되어버린 자식 프로세스

주로 `fork()` 를 이용해서 자식 프로세스(:12)를 생성했을때 사용한다. `wait()` 를 쓰면 자식프로세스가 종료할때까지 해당영역에서 부모프로세스가 `sleep()` 모드로 기다리게 된다. 이는 자식프로세스와 부모프로세스의 동기화를 위한목적으로 부모프로세스가 자식프로세스보다 먼저 종료되어서 자식프로세스가 고아 프로세스(PPID 가 1)인 프로세스가 되는걸 방지하기 위한 목적이다.

만약 자식 프로세스가 종료되었다면 함수는 즉시 리턴되며, 자식이 사용한 모든 시스템자원을 해제한다.

← `wait()` 함수를 통해, 자식 프로세스의 descriptor와 body 모두 제거됨

그런데 어떤이유로 부모가 `wait()`를 호출하기 전에 자식 프로세스가 종료버리는 경우도 있다(잘못된 메모리 연산등으로 인한 죽음, 혹은 정상적으로), 이럴경우 자식프로세스는 좀비(:12)프로세스가 되는데, `wait()`함수는 즉시 리턴하도록 되어있다.

`wait()`의 인자 `status` 를 통하여 자식 프로세스의 상태를 받아올수 있는데, 자식프로세스의 상태값은 자식프로세스의 종료값 * 256(FF) 이다.

인수	<code>int status</code>	자식 프로세스 종료 상태
반환	<code>pid_t</code>	종료된 자식 프로세스 ID

예제

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

int main()
{
    int counter = 1;
    int status;
    pid_t pid;
    pid_t pid_child;

    pid = fork();

    switch( pid)
    {
        case -1 :
        {
            printf( "자식 프로세스 생성 실패\n");
            return -1;
        }
        case 0 :
        {
            printf( "저는 자식 프로세스로 5까지 카운트하고 종료하겠습니다.\n");
            while( 6 > counter )
            {
                printf( "자식: %d\n", counter++);
                sleep( 1);
            }
            return 99;
        }
        default :
        {
            printf( "저는 부모 프로세스로 자식 프로세스 작업이 끝날 때 까지 대기합니다..\n");

            pid_child = wait( &status);
            printf( "종료된 자식 프로세스 ID는 %d이며,", pid_child);
            if ( 0 == ( status & 0xff) )
            {
                printf( "정상적으로 종료되었고 반환값은 %d입니다\n", status >> 8);
            }
            else
            {
                printf( "비 정상으로 종료되었고 종료 시그널 번호는 %d입니다\n", status);
            }
            printf( "이제 제일을 처리하겠습니다.\n");

            while( 1 )
            {
                printf( "부모: %d\n", counter++);
                sleep( 1);
            }
        }
    }
}
```

← 부모 프로세스의 시각점이 `wait()` 함수를 호출함.