

Lab. 2 프로세스 관리

1) 프로세스 생성: *fork()*

: *fork()* 함수를 통해 새 프로세스를 생성하고, 부모-자녀 프로세스 관계를 확인한다. 프로세스가 생성되면 부모의 모든 소스 코드와 변수 등이 포함된 메모리가 100% 자녀 프로세스로 복사된다.

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

static void child()
{
    printf("I'm child! my pid is %d.\n", getpid());
    exit(EXIT_SUCCESS);
}

static void parent(pid_t pid_c)
{
    printf("I'm parent! my pid is %d and the pid of my child is %d.\n",
        getpid(), pid_c);
    exit(EXIT_SUCCESS);
}

int main(void)
{
    pid_t ret;
    ret = fork();
    if (ret == -1)
        err(EXIT_FAILURE, "fork() failed");
    if (ret == 0) {
        // child process came here because fork() returns 0 for child process
        child();
    } else {
        // parent process came here because fork() returns the pid of newly
        // created child process (> 1)
        parent(ret);
    }
    // shouldn't reach here
    err(EXIT_FAILURE, "shouldn't reach here");
}
```

그림 1 fork.c

아래와 같이 컴파일 및 실행한다.

```
$ cc -o fork fork.c
```

```
$ ./fork
```

출력문에서 나온 메시지를 확인하고 부모-자식 관계의 pid 번호도 확인한다.

2) 새 프로세스 생성 및 실행: *fork-and-exec*

: *fork()*와 달리 부모 프로세스의 소스 코드를 물려받지 않고 새 프로그램 내용을 실행하는 방법으로 *fork-and-exec*를 사용한다. *fork-and-exec.c*는 다음과 같은 순서이다.

- a. 프로세스를 새로 만든다.
- b. 부모 프로세스는 echo hello 프로그램을 생성한 뒤 자신의 pid와 자식 프로세스의 pid를 출력하고 종료한다. 자식 프로세스는 자신의 pid를 출력하고 종료한다.

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

static void child()
{
    char *args[] = { "/bin/echo", "hello" , NULL};
    printf("I'm child! my pid is %d.\n", getpid());
    fflush(stdout);
    execve("/bin/echo", args, NULL);
    err(EXIT_FAILURE, "exec() failed");
}

static void parent(pid_t pid_c)
{
    printf("I'm parent! my pid is %d and the pid of my child is %d.\n",
        getpid(), pid_c);
    exit(EXIT_SUCCESS);
}

int main(void)
{
    pid_t ret;
    ret = fork();
    if (ret == -1)
        err(EXIT_FAILURE, "fork() failed");
    if (ret == 0) {
        // child process came here because fork() returns 0 for child process
        child();
    } else {
        // parent process came here because fork() returns the pid of newly
        // created child process (> 1)
        parent(ret);
    }
    // shouldn't reach here
    err(EXIT_FAILURE, "shouldn't reach here");
}
```

그림 2 fork-and-exec.c

컴파일 후 실행한다.

```
$ cc -o fork-and-exec fork-and-exec.c
./fork-and-exec
```

위에서 설명한대로 결과가 나왔는지 확인하고 분석한다.