

---

# Process Control

---

## System Programming

## gcc 사용하기

### □ 우분투 계정 추가 및 setuid 테스트

### □ 책자(이론) 예제 코드 컴파일 및 실행 해보기

- Prog. 7. 4
  - 소스 코드를 컴파일하여 “**echoarg**”로 저장하기
  - /home/sjhong(계정)/bin폴더로 echoarg 파일 복사 (sudo cp echoarg /home/sjhong/bin)
- testinterp파일 생성하기
  - /home/sjhong/bin폴더에서 gedit testinterp로 생성
  - 생성 후, chmod u+x testinterp로 실행권한을 주기
- Prog. 8. 20
  - testinterp파일 생성 후, 코드를 컴파일해서 실행해보기
- Prog. 8.22 & 8.23
  - 2개 소스를 같이 컴파일하기 (예 : gcc -o prog.8.22 prog.8.22c prog.8.23.c.... (기타 옵션 생략))
  - Shell에서 직접 입력한 것과 결과 비교해보기

## Process Control

### □ 우분투 계정 추가하기

- Root계정(su명령 사용)에서 adduser로 계정 추가하기  
(필자의 경우 계정이름: wmlab)

```
root@ubuntu: /home/sjhong
sjhong@ubuntu:~$ su
Password:
root@ubuntu:/home/sjhong# adduser wmlab
Adding user `wmlab' ...
Adding new group `wmlab' (1002) ...
Adding new user `wmlab' (1002) with group `wmlab' ...
Creating home directory `/home/wmlab' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
Sorry, passwords do not match
passwd: Authentication token manipulation error
passwd: password unchanged
Try again? [y/N] y
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for wmlab
Enter the new value, or press ENTER for the default
    Full Name []: WMLAB
    Room Number []: 1000
    Work Phone []: 01011111111
    Home Phone []: 01022222222
    Other []:
Is the information correct? [Y/n] y
root@ubuntu:/home/sjhong#
```

## Process Control

### □ 우분투 계정 추가하기

- 아래와 같이 계정 확인하기

```
sjhong@ubuntu: ~/sysprog
sjhong@ubuntu:~/sysprog$ whoami
sjhong
sjhong@ubuntu:~/sysprog$ gedit /etc/passwd
sjhong@ubuntu:~/sysprog$ tail /etc/passwd
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
hplip:x:115:7:HPLIP system user,,,:/var/run/hplip:/bin/false
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/bin/false
pulse:x:117:124:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false
saned:x:119:127::/var/lib/saned:/bin/false
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false
sjhong:x:1000:1000:seokjoon,,,:/home/sjhong:/bin/bash
apue:x:1001:1001:Sjhong,2000,0222201088,01022537901:/home/apue:/bin/bash
wmlab:x:1002:1002:WMLAB,1000,0101111111,0102222222:/home/wmlab:/bin/bash
sjhong@ubuntu:~/sysprog$
```

## Process Control

### ❑ Setuid 테스트 코드 작성(ex\_setuid.c)

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

int main(int argc, char **argv)
{
    if(argc != 2)
    {
        printf("Usage:%s [user id] \n", argv[0]);
        printf("*** user id is number ***\n");
        return 1;
    }

    printf("real user id :%d effective user id :%d\n", getuid(), geteuid());
    uid_t uid = atoi(argv[1]);
    setuid(uid);
    printf("real user id:%d effective user id:%d\n", getuid(), geteuid());
}
```

## Process Control

### ❑ Setuid 테스트 코드 (ex\_setuid.c)

- gcc -o ex\_setuid ex\_setuid.c로 컴파일 후
- 아래와 같이 ex\_setuid 파일에 set user id 비트를 셋팅하고
- 새로운 계정(wmlab)으로 이동하여 아래와 같은 명령어를 입력하고 결과를 확인한다.
- setuid를 통해서 유효아이디 (effective)가 바뀐 것을 확인할 수 있다.
  - Super user(0)로는 바뀌지 않음을 확인할 수 있다.

```
wmlab@ubuntu: /home/sjhong/sysprog
sjhong@ubuntu:~/sysprog$ chmod u+s ex_setuid
sjhong@ubuntu:~/sysprog$ su -l wmlab
Password:
wmlab@ubuntu:~$ cd /home/sjhong/sysprog/
wmlab@ubuntu:/home/sjhong/sysprog$ ./ex_setuid 1002
real user id :1002 effective user id :1000
real user id:1002 effective user id:1002
wmlab@ubuntu:/home/sjhong/sysprog$ ./ex_setuid 1000
real user id :1002 effective user id :1000
real user id:1002 effective user id:1000
wmlab@ubuntu:/home/sjhong/sysprog$ ./ex_setuid 0
real user id :1002 effective user id :1000
real user id:1002 effective user id:1000
wmlab@ubuntu:/home/sjhong/sysprog$
```

## Process Control

### ❑ Setuid 테스트 코드 (ex\_setuid.c)

- 다시 ex\_setuid 파일에 set user id 비트를 해제하고
- 새로운 계정(wmlab)으로 이동하여 아래와 같은 명령어를 입력하고 결과를 확인한다.
- setuid를 호출해도 real user id와 effective id가 모두 바뀌지 않은 것을 알 수 있다.

```
wmlab@ubuntu: /home/sjhong/sysprog
sjhong@ubuntu:~/sysprog$ chmod u-s ex_setuid
sjhong@ubuntu:~/sysprog$ su -l wmlab
Password:
wmlab@ubuntu:~$ cd /home/sjhong/sysprog/
wmlab@ubuntu:/home/sjhong/sysprog$ ./ex_setuid 1002
real user id :1002 effective user id :1002
real user id:1002 effective user id:1002
wmlab@ubuntu:/home/sjhong/sysprog$ ./ex_setuid 1000
real user id :1002 effective user id :1002
real user id:1002 effective user id:1002
wmlab@ubuntu:/home/sjhong/sysprog$ ./ex_setuid 0
real user id :1002 effective user id :1002
real user id:1002 effective user id:1002
wmlab@ubuntu:/home/sjhong/sysprog$
```

## Process Control

### ❑ Setuid 테스트 코드 (ex\_setuid.c)

- 마지막으로 su 명령으로 root 계정으로 들어가서
- 아래와 같이 id를 바꿀 것을 요청하면, real id와 effective id 모두 바뀌는 것을 알 수 있다.

```
root@ubuntu:/home/sjhong/sysprog# ./ex_setuid 1002
real user id :0 effective user id :0
real user id:1002 effective user id:1002
root@ubuntu:/home/sjhong/sysprog#
```



## Process Environment

### □ Prog. 7.4 코드

```
#include "apue.h"

int
main(int argc, char *argv[])
{
    int    i;

    for (i = 0; i < argc; i++)    /* echo all command-line args */
        printf("argv[%d]: %s\n", i, argv[i]);
    exit(0);
}
```

**Figure 7.4** Echo all command-line arguments to standard output

## Process Control

### ❑ testinterp 파일 생성

- `#!/home/sjhong/bin/echoarg foo`

## Process Control

### □ Prog 8.20 코드

```
#include "apue.h"
#include <sys/wait.h>

int
main(void)
{
    pid_t    pid;

    if ((pid = fork()) < 0) {
        err_sys("fork error");
    } else if (pid == 0) { /* child */
        if (execl("/home/sar/bin/testinterp",
                  "testinterp", "myarg1", "MY ARG2", (char *)0) < 0)
            err_sys("execl error");
    }
    if (waitpid(pid, NULL, 0) < 0) /* parent */
        err_sys("waitpid error");
    exit(0);
}
```

자신이 만든 경로

Figure 8.20 A program that execs an interpreter file

## Process Control

## □ Prog 8.22 코드

```

#include    <sys/wait.h>
#include    <errno.h>
#include    <unistd.h>

int
system(const char *cmdstring)    /* version without signal handling */
{
    pid_t    pid;
    int      status;

    if (cmdstring == NULL)
        return(1);    /* always a command processor with UNIX */

    if ((pid = fork()) < 0) {
        status = -1;    /* probably out of processes */
    } else if (pid == 0) {    /* child */
        execl("/bin/sh", "sh", "-c", cmdstring, (char *)0);
        _exit(127);    /* execl error */
    } else {    /* parent */
        while (waitpid(pid, &status, 0) < 0) {
            if (errno != EINTR) {
                status = -1; /* error other than EINTR from waitpid() */
                break;
            }
        }
    }

    return(status);
}

```

Figure 8.22 The system function, without signal handling

## Process Control

## □ Prog 8.23 코드

```
#include "apue.h"
#include <sys/wait.h>

int
main(void)
{
    int      status;

    if ((status = system("date")) < 0)
        err_sys("system() error");

    pr_exit(status);

    if ((status = system("nosuchcommand")) < 0)
        err_sys("system() error");

    pr_exit(status);

    if ((status = system("who; exit 44")) < 0)
        err_sys("system() error");

    pr_exit(status);

    exit(0);
}
```

Figure 8.23 Calling the system function

## Process Control

### □ Prog. 8.20 실행

```
$ cat /home/sar/bin/testinterp  
#!/home/sar/bin/echoarg foo  
$ ./a.out  
argv[0]: /home/sar/bin/echoarg  
argv[1]: foo  
argv[2]: /home/sar/bin/testinterp  
argv[3]: myarg1  
argv[4]: MY ARG2
```

자신이 만든 경로

## Process Control

### □ Prog. 8. 22 & 23 실행

```
$ ./a.out
```

```
Sat Feb 25 19:36:59 EST 2012
```

```
normal termination, exit status = 0
```

*for date*

```
sh: nosuchcommand: command not found
```

```
normal termination, exit status = 127
```

*for nosuchcommand*

```
sar          console   Jan  1 14:59
```

```
sar          ttys000    Feb  7 19:08
```

```
sar          ttys001    Jan 15 15:28
```

```
sar          ttys002    Jan 15 21:50
```

```
sar          ttys003    Jan 21 16:02
```

```
normal termination, exit status = 44
```

*for exit*

---

*Thank you for your attention !!*

---

Q and A