Prcocess Environment

System Programming

실습

gcc 사용하기

- ❑ **지난번 파일 입출력 함수로 작업했던 코드를 표준 입출력 함수로 바꿔보기**
  – 예제코드를 수정해서 테스트해보기
  – fopen, fgets, fputs, fclose를 사용 (open, read, write, close 사용 X)
- ❑ **Helloworld 컴파일해서 ldd로 확인해보기**
  – static 옵션으로 컴파일 한 경우와 차이 확인하기
- ❑ **setjmp/longjmp 코드 작성 및 테스트해보기**
  – 추가 실습 : longjmp함수를 get_token함수에서 실행해보기 (val인자값 : 2로 설정)
- ❑ **책자(이론) 예제 코드 컴파일 및 실행 해보기**
  – Prog. 7. 3
  – Prog. 7. 4
  – Prog. 7. 13

예제 코드

Example Code

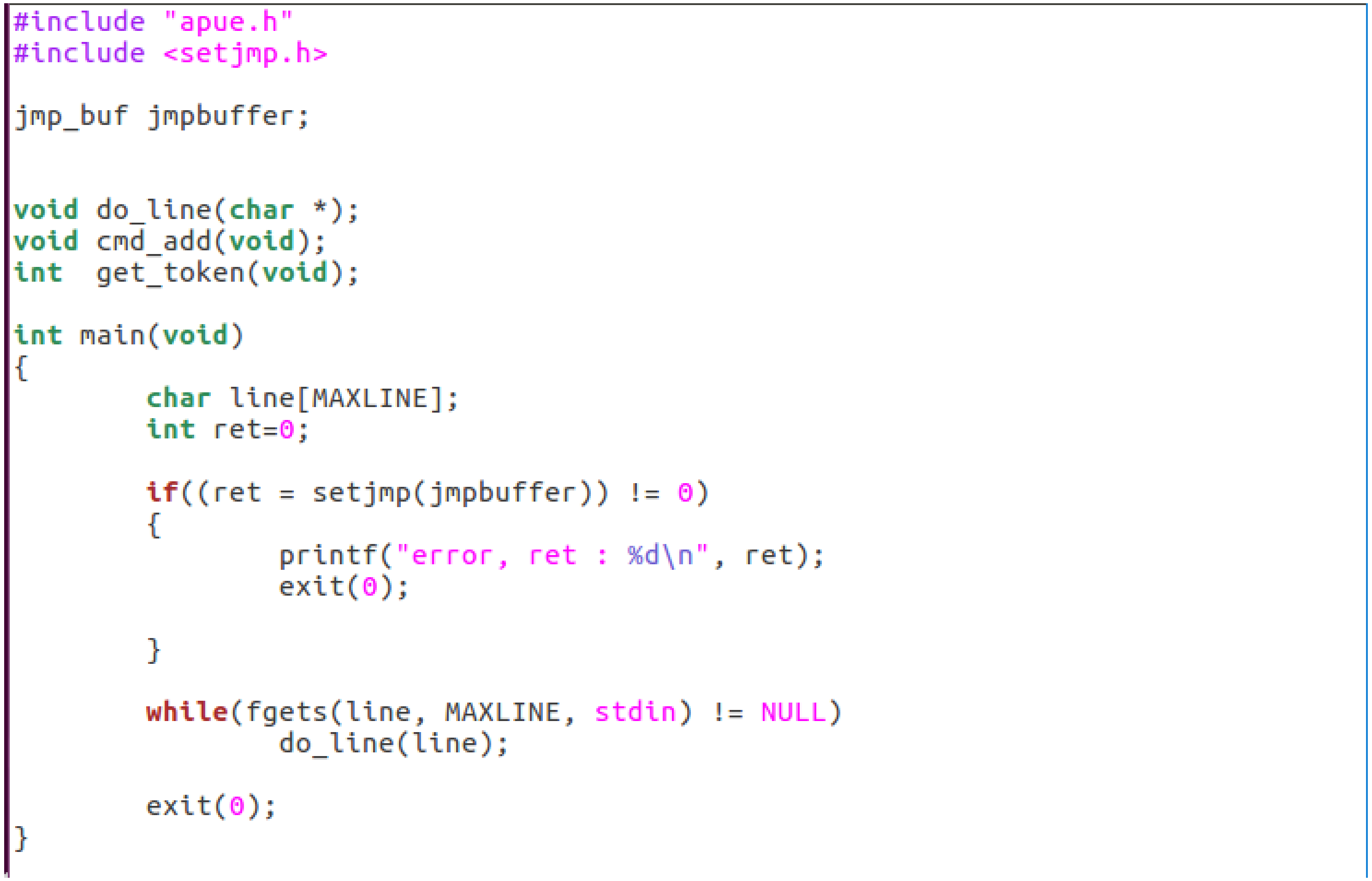
```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>

int main ()
{
        int filedes;
        char tmpstr[] = "Hello my friend!!₩n";
        filedes = open ("ex3_text.txt", O_RDWR|O_CREAT, 0644);
        write (filedes, tmpstr, strlen(tmpstr));

        close (filedes);
        return 0;
}
```

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

int main ()
{
        int filedes;
        ssize_t nread;
        char tmpstr[1024];

        filedes = open ("ex3_text.txt", O_RDWR);
        memset (tmpstr, 0, sizeof(tmpstr));
        nread = read (filedes, tmpstr, 1024);
        printf ("%s", tmpstr);

        close (filedes);
        return 0;
}
```

실습

Process Environment

- setjmp/longjmp 테스트 코드

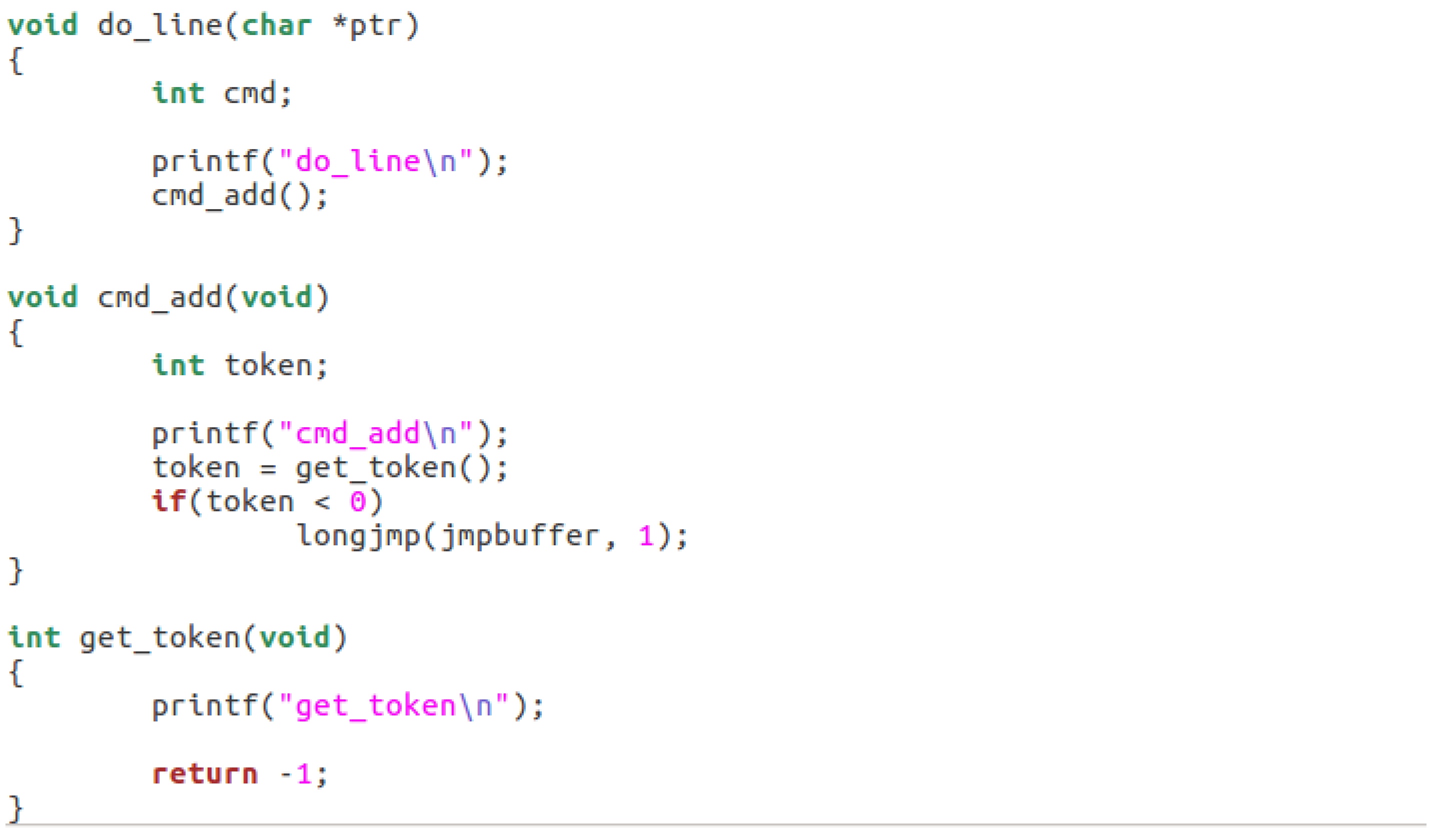


```
#include "apue.h"
#include <setjmp.h>

jmp_buf jmpbuffer;


void do_line(char *);
void cmd_add(void);
int  get_token(void);

int main(void)
{
        char line[MAXLINE];
        int ret=0;

        if((ret = setjmp(jmpbuffer)) != 0)
        {
                printf("error, ret : %d\n", ret);
                exit(0);

        }

        while(fgets(line, MAXLINE, stdin) != NULL)
                do_line(line);

        exit(0);
}
```

실습

Process Environment

- ❑ **setjmp/longjmp 테스트 코드**

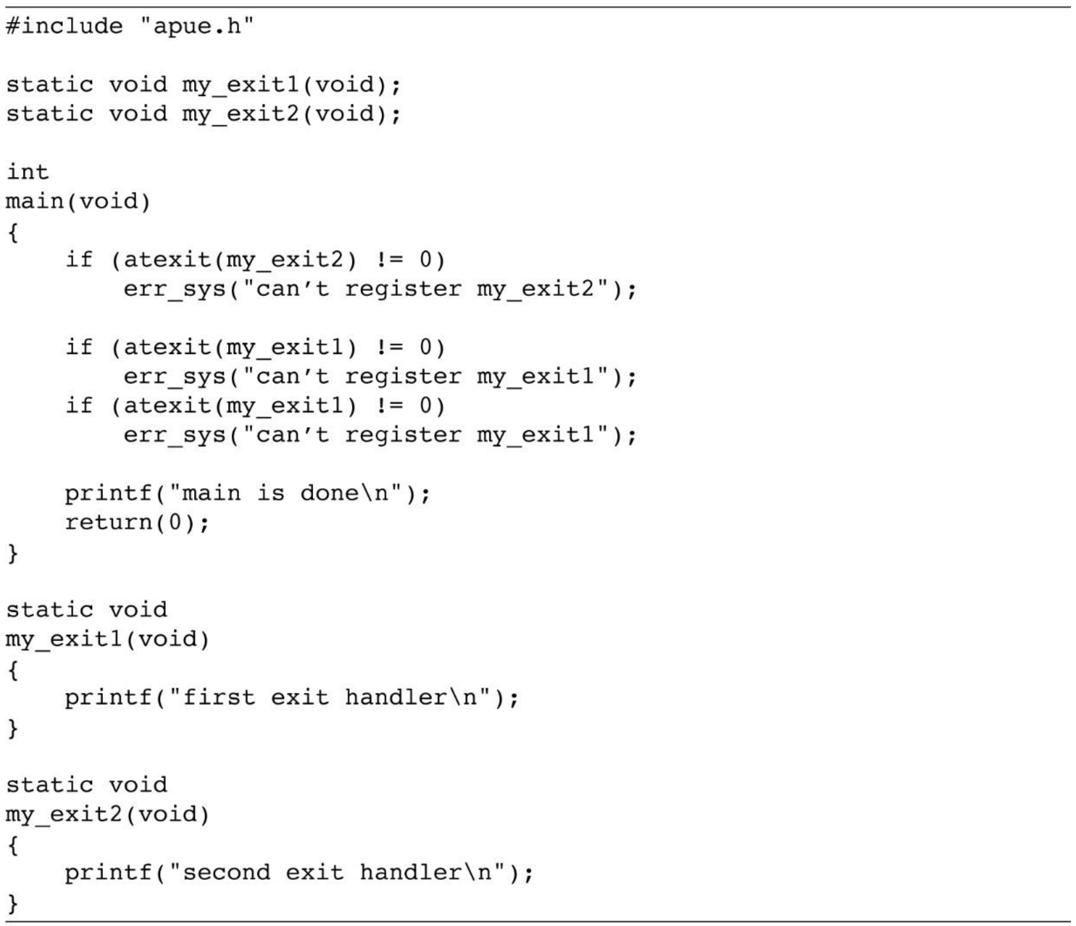


```c
void do_line(char *ptr)
{
        int cmd;

        printf("do_line\n");
        cmd_add();
}

void cmd_add(void)
{
        int token;

        printf("cmd_add\n");
        token = get_token();
        if(token < 0)
                longjmp(jmpbuffer, 1);
}

int get_token(void)
{
        printf("get_token\n");

        return -1;
}
```

실습

Process Environment

❑ Prog. 7. 3 코드



```
#include "apue.h"

static void my_exit1(void);
static void my_exit2(void);

int
main(void)
{
    if (atexit(my_exit2) != 0)
        err_sys("can't register my_exit2");

    if (atexit(my_exit1) != 0)
        err_sys("can't register my_exit1");
    if (atexit(my_exit1) != 0)
        err_sys("can't register my_exit1");

    printf("main is done\n");
    return(0);
}

static void
my_exit1(void)
{
    printf("first exit handler\n");
}

static void
my_exit2(void)
{
    printf("second exit handler\n");
}
```

Figure 7.3 Example of exit handlers

실습

Process Environment

❑ Prog. 7. 4 코드

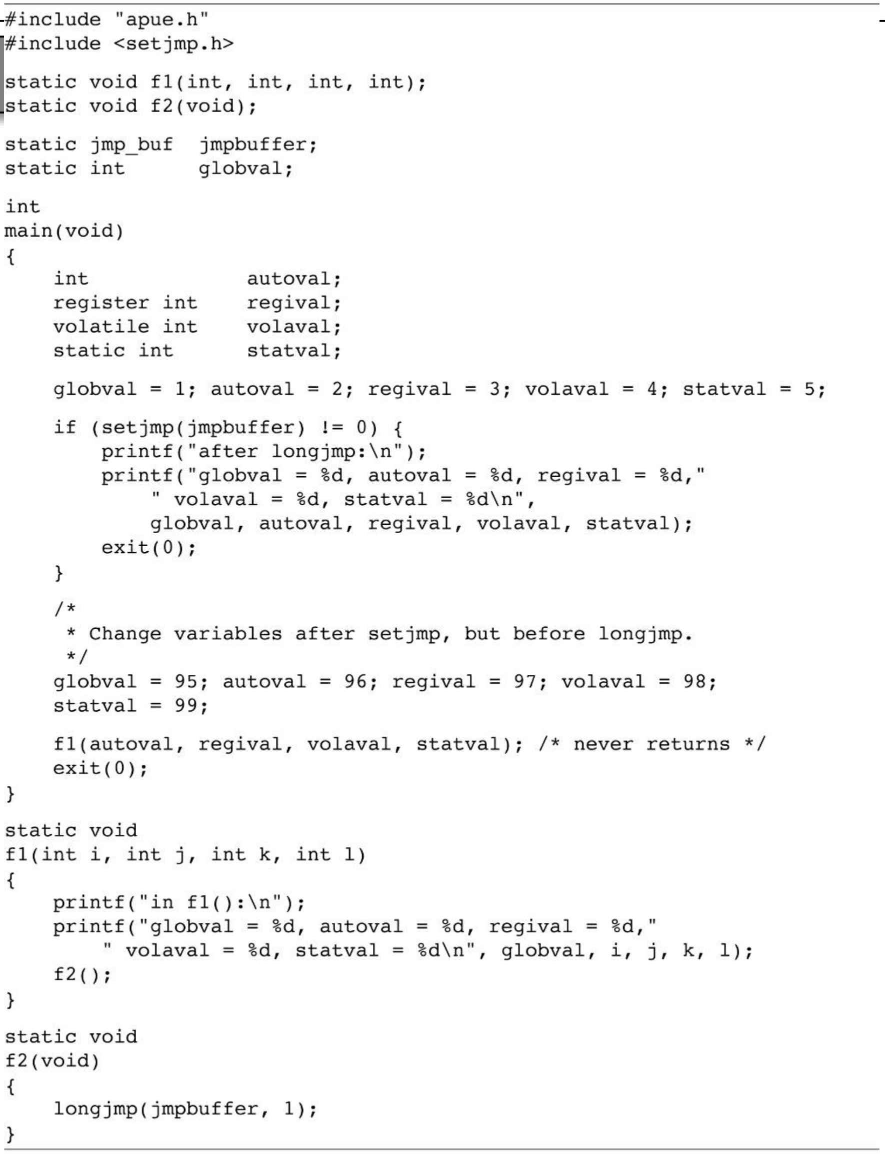
```
#include "apue.h"

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

실습

Process Environment

- ❑ **Prog. 7. 13 코드**



```
#include "apue.h"
#include <setjmp.h>

static void f1(int, int, int, int);
static void f2(void);

static jmp_buf  jmpbuffer;
static int      globval;

int
main(void)
{
    int             autoval;
    register int    regival;
    volatile int    volaval;
    static int      statval;

    globval = 1; autoval = 2; regival = 3; volaval = 4; statval = 5;

    if (setjmp(jmpbuffer) != 0) {
        printf("after longjmp:\n");
        printf("globval = %d, autoval = %d, regival = %d,"
            " volaval = %d, statval = %d\n",
            globval, autoval, regival, volaval, statval);
        exit(0);
    }

    /*
     * Change variables after setjmp, but before longjmp.
     */
    globval = 95; autoval = 96; regival = 97; volaval = 98;
    statval = 99;

    f1(autoval, regival, volaval, statval); /* never returns */
    exit(0);
}

static void
f1(int i, int j, int k, int l)
{
    printf("in f1():\n");
    printf("globval = %d, autoval = %d, regival = %d,"
        " volaval = %d, statval = %d\n", globval, i, j, k, l);
    f2();
}

static void
f2(void)
{
    longjmp(jmpbuffer, 1);
}
```

Figure 7.13 Effect of `longjmp` on various types of variables

실습

Process Environment

❑ **Prog. 7. 3 실행**

```
$ ./a.out
main is done
first exit handler
first exit handler
second exit handler
```

실습

Process Environment

❑ **Prog. 7. 4 실행**

```
$ ./echoarg arg1 TEST foo
argv[0]: ./echoarg
argv[1]: arg1
argv[2]: TEST
argv[3]: foo
```

실습

Process Environment

❑ **Size 확인**

```
$ size /usr/bin/cc /bin/sh
   text    data     bss     dec     hex  filename
 346919    3576    6680  357175   57337  /usr/bin/cc
 102134    1776   11272  115182   1c1ee  /bin/sh
```

실습

Process Environment

❑ **Gcc without shared libraries**

```
$ gcc -static hello1.c
$ ls -l a.out
-rwxr-xr-x  1 sar     879443 Sep 2 10:39 a.out
$ size a.out
   text    data     bss     dec     hex  filename
 787775    6128   11272  805175   c4937  a.out
```

prevent gcc from using shared libraries

실습

Process Environment

❑ **Gcc with shared libraries**

gcc defaults to use shared libraries

```
$ gcc hello1.c
$ ls -l a.out
-rwxr-xr-x  1 sar        8378 Sep 2 10:39 a.out
$ size a.out
   text     data     bss      dec      hex   filename
   1176      504      16     1696      6a0   a.out
```

실습

Process Environment

❑ Prog. 7. 13 실행

```
$ gcc testjmp.c                     compile without any optimization
$ ./a.out
in f1():
globval = 95, autoval = 96, regival = 97, volaval = 98, statval = 99
after longjmp:
globval = 95, autoval = 96, regival = 97, volaval = 98, statval = 99
$ gcc -O testjmp.c                  compile with full optimization
$ ./a.out
in f1():
globval = 95, autoval = 96, regival = 97, volaval = 98, statval = 99
after longjmp:
globval = 95, autoval = 2, regival = 3, volaval = 98, statval = 99
```

Thank you for your attention !!

Q and A