TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – GJ (박현우)
uc820@naver.com

1. 시스템 프로그래밍 – 5 파일입출력

```c
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include<dirent.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<string.h>

void recursive_dir(char *dname);

int main(int argc, char **argv){
    recursive_dir(".");
    return 0;
}

void recursive_dir(char *dname){

    struct dirent *p; // directory entry
    struct stat buf;  // status
    DIR *dp;          //directory
    chdir(dname);
    dp = opendir(".");
    printf("\t%s: \n", dname); // 디렉토리 이름
    while(p = readdir(dp))
        printf("%s\n", p->d_name);// 하위 파일 이름
    rewinddir(dp);  // 다시 directory position 초기로 복귀
    while(p = readdir(dp)){
        stat(p->d_name, &buf);
        if(S_ISDIR(buf.st_mode))
            if(strcmp(p->d_name, ".") && strcmp(p->d_name, "..")) // && shorcut  ,  "."이면 0
                recursive_dir(p->d_name); // 다른 디렉토리
    }
    chdir(".."); // change dir  .. 상위 디렉토리,,   프로그램 내부에서만 이루어지는 작업
    closedir(dp);
}
```

```c
#include<unistd.h>
#include<stdio.h>
#include<errno.h>
#include<stdlib.h>

int global = 100;

int main(void){

    int local = 10;
    pid_t pid;
    int i;
    pid = fork(); //  fork()로 밑에 코드를 복사하고 또다른 프로세스를 만듦

    if(pid>0){    // parent는 0보다 크고
        printf("global : %d, local : %d\n", global, local);
    }
    else if(pid == 0){  // 자식은 0이다.
        global++;
        local++;
        printf("global : %d, local : %d\n", global, local);
    }else{

        perror("fork() ");
        exit(-1);
    }
    printf("\n");
    return 0;
}
```

1. 시스템 프로그래밍 – 5 파일입출력

```c
#include<unistd.h>
#include<stdio.h>
#include<errno.h>
#include<stdlib.h>
#include<fcntl.h>

int main(void){

    int fd,ret;
    char buf[1024];
    pid_t pid;
    fd = open("myfifo", O_RDWR);  // myfifo라는 파이프를 읽고 쓰기 가능한 모드로 열고
    if(((pid=fork()) > 0)){   // 자식 프로세스를 만든다.

        for(;;){

            ret = read(0,buf, sizeof(buf));
            buf[ret] = 0;
            printf("Keyboard: %s\n",buf);
        }
    }else if(pid == 0){

        for(;;){
            ret = read(fd, buf, sizeof(buf));  // 블로킹 함수이지만, myfifo에서의 입력이 들어오면
            buf[ret] = 0;                      // 자식 프로세스에서 fd만 보고 있기 때문에
                                               // 논 블로킹 함수처럼 동작함.
            printf("myfifo: %s\n",buf);
        }
    }
    else{
        perror("fork() ");
        exit(-1);
    }

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

```c
#include<unistd.h>
#include<stdio.h>
#include<errno.h>
#include<stdlib.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/wait.h>

int main(void){

    pid_t pid;
    int status;
    if( (pid = fork()) > 0 ){

        wait(&status);
        printf("status: 0x%x\n",(status >> 8) & 0xff);  // status를 8비트 오른쪽으로 쉬프트하고
    }else if(pid == 0){                                 // 1111 1111을 and 연산해서 앞의 8개의 비트만
                                                        // 확인한다.
        exit(7);        // 자식 프로세스의 상태가 7로 return 받기 때문에 위에 status는 7이 나온다.

    }else{

        perror("fork() ");
        exit(-1);
    }
    return 0;
}
```

1. 시스템 프로그래밍 – 5 (ls – alR 구현하기 코드)

```c
void recursive_dir(char *dname, int flag){

    DIR *dp;
    int i = 0;
    struct dirent *p;
    struct stat buf;

    if( flag & C){
        chdir(dname);
        printf("%s: \n",dname);
    }

    dp = opendir(".");

    while(p = readdir(dp)){

        if( !(flag & A)){
            if(p->d_name[0] == '.')
                continue;
        }
        if ( flag & B){
            file_att(p->d_name);
        }
        printf("%-16s ", p->d_name);


        if( (i + 1) % 5 == 0 && !(flag & B) )
            printf("\n");
        else if( flag & B)
            printf("\n");


        i++;
    }

    printf("\n");
    printf("\n");

    if (flag & C){
        rewinddir(dp);  // 다시 directory position 초기로 복귀
        while(p = readdir(dp)){
            stat(p->d_name, &buf);
            if(S_ISDIR(buf.st_mode))
                if(strcmp(p->d_name, ".") && strcmp(p->d_name, "..")){
                    recursive_dir(p->d_name, flag);
                }
        }
        chdir("..");
    }
    printf("\n");
    closedir(dp);


}
```

```c
void file_att(char *p){
    struct stat buf;
    struct passwd *pw;
    struct group *gr;

   struct tm *tm;
   char ch;
   char perm[11] = "----------";
   char rwx[4] = "rwx";
   char sst[4] = "sst";
   int i;
   stat(p, &buf);

   if(S_ISDIR(buf.st_mode)) // 디렉토리
       perm[0] = 'd';
   if(S_ISREG(buf.st_mode)) // 레귤러 일반 파일
       perm[0] = '-';
   if(S_ISFIFO(buf.st_mode))  // 파이프
       perm[0] = 'p';
   if(S_ISLNK(buf.st_mode))  // 링크 바로가기
       perm[0] = 'l';
   if(S_ISSOCK(buf.st_mode)) // 소켓
       perm[0] = 's';
   if(S_ISCHR(buf.st_mode)) //케릭터 디바이스
       perm[0] = 'c';
   if(S_ISBLK(buf.st_mode)) // 블락 디바이스
       perm[0] = 'b';

   for(i = 0; i<9; i++){

       if( (buf.st_mode >> ( 8 - i) & 1))
           perm[i + 1] = rwx[i%3];

   }

   for(i = 0; i<3; i++){

       if(buf.st_mode >> (11 - i) & 1){
           if(perm[ (i+1) * 3 == '-'])
               perm[ (i + 1) *3 ] = sst[i] ^ 0x20;
           else
               perm[ (i + 1) * 3] = sst[i];
       }
   }

   printf("%s ", perm);
   printf("%d ", buf.st_nlink);
   pw = getpwuid(buf.st_uid);
   printf("%s ", pw->pw_name);
   gr = getgrgid(buf.st_gid);
   printf("%s ",gr->gr_name);
   printf("%d ", buf.st_size);
   tm = localtime(&buf.st_mtime);
   printf("%d-%02d-%02d %02d:%02d ", tm->tm_year + 1900, tm->tm_mon +1, tm->tm_mday, tm->tm_ho
ur, tm->tm_min);

}
```

```c
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include<dirent.h>
#include<sys/stat.h>
#include<string.h>
#include<pwd.h>
#include<grp.h>
#include<time.h>


#define A ( 1 << 0)
#define B ( 1 << 1)
#define C ( 1 << 2)

void file_att(char *p);
void recursive_dir(char *dname, int flag);

int main(int argc, char **argv){

    int cmd, flag = 0;

    while( (cmd = getopt(argc, argv, "alR")) > 0 ){

        switch(cmd){

            case 'a':
                flag |= 1;
                break;
            case 'l':
                flag |= 2;
                break;
            case 'R':
                flag |= 4;
                break;

        }

    }

    recursive_dir(".", flag);

    return 0;
}
```

1. 시스템 프로그래밍 – 5 (ls – alR 구현하기 결과)

```
hyunwoopark@hyunwoopark-P65-P67SG:~/hw$ ./a.out -lR
.:
-rw-rw-r-- 1 hyunwoopark hyunwoopark 242 2018-03-11 14:00 1.c
-rw-rw-r-- 1 hyunwoopark hyunwoopark 712 2018-03-03 17:02 func_pointer.c
-rw-rw-r-- 1 hyunwoopark hyunwoopark 209 2018-03-11 15:26 3.c
-rwxrwxr-x 1 hyunwoopark hyunwoopark 10416 2018-03-11 17:19 debug
-rw-rw-r-- 1 hyunwoopark hyunwoopark 2559 2018-03-09 00:16 avl_tree.c
-rw-rw-r-- 1 hyunwoopark hyunwoopark 2672 2018-03-25 01:11 ls.c
-rw-rw-r-- 1 hyunwoopark hyunwoopark 474 2018-03-11 16:10 5.c
-rw-rw-r-- 1 hyunwoopark hyunwoopark 1650 2018-03-11 18:43 21.c
drwxrwxr-x 7 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 ls_check
-rwxrwxr-x 1 hyunwoopark hyunwoopark 13232 2018-03-09 00:01 avl
-rw-rw-r-- 1 hyunwoopark hyunwoopark 916 2018-03-24 23:51 a.c
-rw-rw-r-- 1 hyunwoopark hyunwoopark 638 2018-03-11 16:31 19.c
-rwxrwxr-x 1 hyunwoopark hyunwoopark 13520 2018-03-25 01:11 a.out


ls_check:
drwxrwxr-x 2 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 e.c
drwxrwxr-x 2 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 d.c
drwxrwxr-x 2 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 f.c
drwxrwxr-x 2 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 a.c
drwxrwxr-x 2 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 b.c


e.c:



d.c:



f.c:
```

```
hyunwoopark@hyunwoopark-P65-P67SG:~/hw$ ./a.out -alR
.:
-rw-rw-r-- 1 hyunwoopark hyunwoopark 242 2018-03-11 14:00 1.c
-rw-rw-r-- 1 hyunwoopark hyunwoopark 712 2018-03-03 17:02 func_pointer.c
-rw-rw-r-- 1 hyunwoopark hyunwoopark 209 2018-03-11 15:26 3.c
-rwxrwxr-x 1 hyunwoopark hyunwoopark 10416 2018-03-11 17:19 debug
-rw-rw-r-- 1 hyunwoopark hyunwoopark 2559 2018-03-09 00:16 avl_tree.c
drwxrwxr-x 3 hyunwoopark hyunwoopark 4096 2018-03-25 01:12 .
-rw------- 1 hyunwoopark hyunwoopark 12288 2018-03-09 00:17 .avl_tree.c.swp
-rw-rw-r-- 1 hyunwoopark hyunwoopark 2672 2018-03-25 01:11 ls.c
-rw-rw-r-- 1 hyunwoopark hyunwoopark 474 2018-03-11 16:10 5.c
-rw-rw-r-- 1 hyunwoopark hyunwoopark 1650 2018-03-11 18:43 21.c
-rw-r--r-- 1 hyunwoopark hyunwoopark 12288 2018-03-25 01:12 .ls.c.swp
drwxrwxr-x 7 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 ls_check
-rwxrwxr-x 1 hyunwoopark hyunwoopark 13232 2018-03-09 00:01 avl
drwxr-xr-x 27 hyunwoopark hyunwoopark 4096 2018-03-25 01:11 ..
-rw-rw-r-- 1 hyunwoopark hyunwoopark 916 2018-03-24 23:51 a.c
-rw-rw-r-- 1 hyunwoopark hyunwoopark 638 2018-03-11 16:31 19.c
-rwxrwxr-x 1 hyunwoopark hyunwoopark 13520 2018-03-25 01:11 a.out


ls_check:
drwxrwxr-x 7 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 .
drwxrwxr-x 2 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 e.c
drwxrwxr-x 2 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 d.c
drwxrwxr-x 3 hyunwoopark hyunwoopark 4096 2018-03-25 01:12 ..
drwxrwxr-x 2 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 f.c
drwxrwxr-x 2 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 a.c
drwxrwxr-x 2 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 b.c


e.c:
drwxrwxr-x 2 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 .
drwxrwxr-x 7 hyunwoopark hyunwoopark 4096 2018-03-25 00:48 ..
```

1. 시스템 프로그래밍 – 5 (파일입출력 & ls – a 기능 구현하기 문제)

문제

a.txt에 아래와 같은 문자열들이 적혀있다.

apple
banana
peach
mango
watermelon
cherry
strawberry

이들을 읽어서 b.txt에
각 행별로 아스키 코드를 합친 숫자값을 적으시오.
그리고 숨김 파일로 .c.txt를 만드시오.
.c.txt에는 각 열별로
아스키 코드 수를 합친 결과값을 적으시오.
이 상태에서 ls -a 기능을 구현하여 모든 파일을 출력하시오.
(모든 동작이 하나의 파일에서 동작하게 구현하시오)

코드

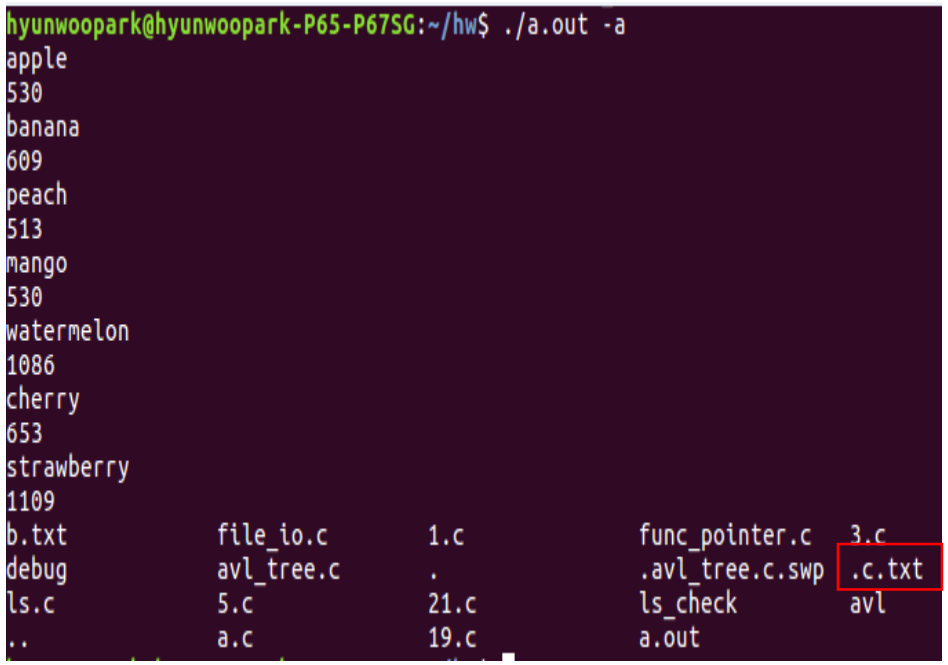
```c
#include<fcntl.h>
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/types.h>
#include<dirent.h>

int init_str(char *str){

    int i;
    for(i =0; str[i];i++){
            str[i] = '\0';
    }
}

void make_ascii(void){

    int fdin, fdout, i;
    char buf[64] = {0};
    int sum;
    char str[64] = {0};
    fdin = open("b.txt", O_RDONLY);
    fdout = open(".c.txt", O_WRONLY|O_CREAT| O_APPEND | O_TRUNC, 0644);

    read(fdin, buf, sizeof(buf));

    for(i = 0; buf[i]; i++){

        printf("%c",buf[i]);

        if( !(strncmp(&buf[i], "\n", 1))){
            printf("%d \n", sum);
            sprintf(str, "%d", sum);
            str[strlen(str)] = '\n';
            write(fdout, str, strlen(str));
            init_str(str);
            sum =0;
        }else{

            sum += buf[i];
        }
    }

    close(fdin);
    close(fdout);
}
```

코드

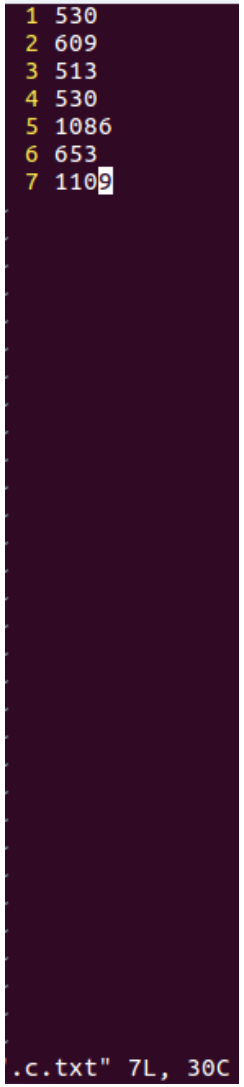
```c
int main(int argc, char **argv){

    DIR *dp;
    struct dirent *p;
    int i=0, cmd;
    int flag =0;
    make_ascii();

    while( (cmd = getopt(argc,argv, "a")) > 0){
        switch(cmd){

            case 'a':
                flag |= 1;
                break;
        }
    }

    dp = opendir(".");
    while(p = readdir(dp)){

        if( !(flag & 1)){
            if(p->d_name[0] == '.')
                continue;
        }
        printf("%-16s ", p->d_name);
        if( (i+1) % 5 == 0)
            printf("\n");

        i++;
    }

    printf("\n");
    closedir(dp);

    return 0;
}
```

1. 시스템 프로그래밍 – 5 (파일입출력 & ls – a 기능 구현하기 문제)

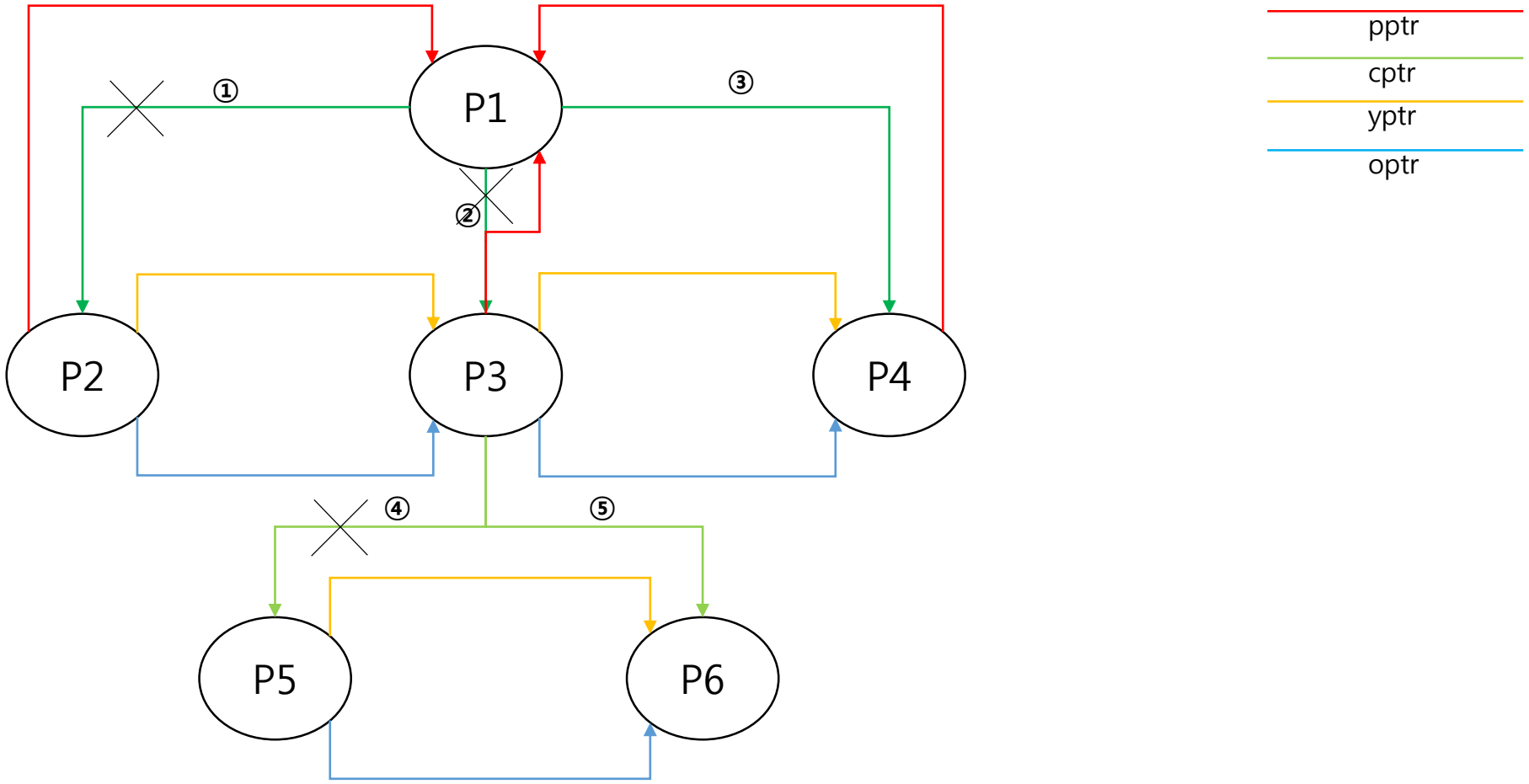
./a.out – a 결과



.c.txt 결과



1. 시스템 프로그래밍 – 5 (Thread)



P1 프로세스는 자식이 생길 때 마다 마지막 프로세스로 cptr을 옮긴다. ① -> ③

P3 프로세스 역시 새로운 자식이 생길 때 마다 cptr을 옮긴다. ④ -> ⑤