Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 전문가 과정

<리눅스 시스템 프로그래밍, 네트워크, 커널>

2018.04.17 - 2 개월차 시험 오답정리

(코 드)

강사 - 이상훈

gcccompil3r@gmail.com

학생 - 안상재

sangjae2015@naver.com

1. 파이프 통신을 구현하고 c type.c라고 입력할 경우, 현재 위치의 디렉토리에 type.c 파일을 생성하도록 프로그래밍 하시오.

```
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>

int flag;

char *check_text(char *text)     // “c type.c” 파일명인지 판별함
{
        int i;
        static char filename[1024];
        int text_len = strlen(text);
        if(text[0] != 'c' && text[1] != ' ')
                return NULL;
        if(text[text_len - 1] != 'c' && text[text_len - 2] != '.')
                return NULL;
        for(i = 2; i < text_len - 2; i++)
        {
                if(text[i] == ' ' || text[i] == '\t')
                        return NULL;
                filename[i - 2] = text[i];
        }
        strcat(filename, ".c");
        return filename;
}

int main(void)
{
        int fo;
        int fd, ret;
        char buf[1024];
        char *string = NULL;
        fd = open("myfifo", O_RDWR);
        fcntl(0, F_SETFL, O_NONBLOCK);
        fcntl(fd, F_SETFL, O_NONBLOCK);
        for(;;)
        {
                if((ret = read(0, buf, sizeof(buf))) > 0)   // 키보드로부터 입력을 받음
                {
                        buf[ret - 1] = 0;
                        printf("Keyboard Input : [%s]\n", buf);
                        string = check_text(buf);          // “c type.c” 을 입력받으면 “type.c” 를 반환함
                        printf("String : %s\n", string);
                }
                if((ret = read(fd, buf, sizeof(buf))) > 0)  // 파이프 통신으로부터 nonblocking으로 입력받음
                {
                        buf[ret - 1] = 0;
                        printf("Pipe Input : [%s]\n", buf);
                        string = check_text(buf);
                        printf("String : %s\n", string);
                }
                fo = open(string, O_WRONLY | O_CREAT | O_TRUNC, 0644); // “type.c” 파일 생성
                close(fo);
        }
        close(fd);
        return 0;
```

```
}
```

2. 369 게임을 작성하시오. 2 초내에 값을 입력하게 하시오. 박수를 쳐야 하는 경우를 Ctrl + C 를 누르도록 한다. 2 초내에 값을 입력하지 못할 경우 게임이 오버되게 한다. Ctrl + C 를 누르면 "Clap!"이라는 문자열이 출력되게 한다.

```
/* gcc 2.c -lm */

#include <stdio.h>
#include <signal.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>

char buf[1024];
int starting_val = 1;
int needs_clap;

void gameover(int signo)
{
    printf("\nGame Over!!!\n");
    exit(0);
}

void clap(void)
{
    printf("\nClap!\n");
    needs_clap--;
}

int find_parm(void)
{
    int i;
    int parm = 0;
    for(i = 0; ; i++)
        if(starting_val < pow(10, i))
            break;

    return parm = i;
}

void counting_clap(int signo)
{
    int i;
    int flag;
    int buf_len;

    if(needs_clap == 0)
    {
        flag = 1;
        buf_len = find_parm();
        for(i = 0; i < buf_len; i++)
        {
            if(i == 0)
                buf[i] = starting_val % (int)pow(10, i + 1);
            else
            {
                buf[i] = starting_val - buf[0];
                buf[i] /= pow(10, i);
```

```
            }
        }
        starting_val++;
    }

    if(flag == 1)
        for(i = 0; i < buf_len; i++)
            if(buf[i] == 3 || buf[i] == 6 || buf[i] == 9)
                needs_clap++;

    if(needs_clap)
    {
        flag = 0;
        clap();
    }
    else
        gameover(0);
}

int main(void)
{
    int i;
    int ret;
    int cmp;
    int buf_len;

    signal(SIGALRM, gameover);
    signal(SIGINT, counting_clap);

    for(;;)
    {
        alarm(2);
        ret = read(0, buf, sizeof(buf));
        buf[ret - 1] = 0;
        buf_len = find_parm();

        for(i = 0; i < buf_len; i++)
            if(buf[i] == '3' || buf[i] == '6' || buf[i] == '9')
                gameover(0);

        cmp = atoi(buf);

        if(starting_val != cmp)
            gameover(0);
        starting_val++;
    }

    return 0;
}
```

71. 인자로 파일을 입력 받아 해당 파일의 앞 부분 5줄을 출력하고 추가적으로 뒷 부분의 5줄을 출력하는 프로그램을 작성하시오.

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

int save_area[1024];

int main(int argc, char **argv)
{
    int i;
    int ret;
    int index;
    int fd;
    char buf[1024];
    fd = open(argv[1], O_RDONLY);
    ret = read(fd, buf, sizeof(buf));   // open 한 파일의 데이터를 모두 read함

    for(i = 0, index = 1; buf[i]; i++)
    {
        if(buf[i] == '\n')
        {
            save_area[index] = i;  /* 개행이 있을 때마다 몇 번째 개행인지, 해당 개행에서의 데이터갯수는 몇개가
                                      되는지 기록함 */
            index++;
        }
    }

    printf("Front 5 Lines\n");
    write(1, buf, save_area[5] + 1);
    printf("Back 5 Lines\n");
    printf("%s", &buf[save_area[index - 7] + 1]);
    return 0;
}
```

83. 디렉토리를 만드는 명령어는 mkdir 명령어다. man -s2 mkdir 을 활용하여 mkdir System Call 을 볼 수 있다. 이를 참고하여 디렉토리를 만드는 프로그램을 작성해보자!

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

int main(int argc, char *argv[])
{
    if(argc != 2)
    {
        printf("Usage: exe dir_name\n");
        exit(-1);
    }

    mkdir(argv[1], 0755);     // mkdir은 디렉토리를 만드는 함수
    return 0;
}
```

84. 이번에는 랜덤한 이름(길이도 랜덤)을 가지도록 디렉토리를 3개 만들어보자! (너무 길면 힘드니까 적당한 크기로 잡도록 함)

```
#include <sys/stat.h>
#include <sys/types.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <time.h>

char *rand_name(void)
{
	int i, len;
	static char buf[8] = "\0";

	memset(buf, 0, sizeof(buf));
	len = rand() % 4 + 4;		// 4~7 글자 수

	for(i = 0; i < len; i++)
		buf[i] = rand() % 26 + 97;  // 파일명을 len 갯수만큼의 소문자로 함

	printf("buf = %s\n", buf);
	return buf;
}

void make_rand_dname(char **dname)
{
	int i;

	for(i = 0; i < 3; i++)
	{
		dname[i] = (char *)malloc(sizeof(char) * 8); // dname[] 각각의 원소에 8글자의 메모리 공간을 확보함
		strcpy(dname[i], rand_name()); // 4~7 글자 수의 파일명을 dname[] 각각의 원소에 복사함
	}
}

int main(void)
{
	int i;
	char *dname[3] = {0};

	srand(time(NULL));
	make_rand_dname(dname);

	for(i = 0; i < 3; i++)
	{
		printf("dname[%d] = %s\n", i, dname[i]);
		mkdir(dname[i], 0755);
	}

	return 0;
}
```

85. 랜덤한 이름을 가지도록 디렉토리 3개를 만들고 각각의 디렉토리에 5 ~ 10개 사이의 랜덤한 이름(길이도 랜덤)을 가지도록 파일을 만들어보자! (너무 길면 힘드니까 적당한 크기로 잡도록함)

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

char *rand_name(void)
{
	int i, len;
	static char buf[8] = "\0";

	memset(buf, 0, sizeof(buf));
	len = rand() % 4 + 4;

	for(i = 0; i < len; i++)
		buf[i] = rand() % 26 + 97;

	printf("buf = %s\n", buf);

	return buf;
}

void make_rand_dname(char **dname)
{
	int i;

	for(i = 0; i < 3; i++)
	{
		dname[i] = (char *)malloc(sizeof(char) * 8);
		strcpy(dname[i], rand_name());
	}
}

void make_rand_file(void)
{
	int i, j, fd, len, cnt;
	char buf[11] = "\0";

	len = rand() % 6 + 5;
	cnt = rand() % 4 + 2;

	for(i = 0; i < cnt; i++)	// buf 파일을 파일명을 바꾸어서 여러개 만들기 위함
	{
		for(j = 0; j < len; j++)
			buf[j] = rand() % 26 + 97;	// buf 파일의 파일명을 정함

		fd = open(buf, O_CREAT, 0644);	// buf 파일을 생성함
		close(fd);

		memset(buf, 0, sizeof(buf));	// buf 배열을 0으로 초기화함
	}
}
```

```
void lookup_dname(char **dname)
{
	int i;

	for(i = 0; i < 3; i++)
	{
		chdir(dname[i]);   // dname[] 파일을 현재 디렉토리로 함(dname[] 파일로 들어감)
		make_rand_file();    // dname[i] 디렉토리로 들어가서 파일을 여러개 만듬
		chdir("..");          // 상위 디렉토리로 나옴
	}
}

int main(void)
{
	int i;
	char *dname[3] = {0};

	srand(time(NULL));
	make_rand_dname(dname);

	for(i = 0; i < 3; i++)
		mkdir(dname[i], 0755);

	lookup_dname(dname);

	return 0;
}
```

86. 85 번까지 진행된 상태에서 모든 디렉토리를 순회하며 3 개의 디렉토리와 그 안의 모든 파일들의 이름 중 a, b, c 가 1개라도 들어있다면 이들을 출력하라! 출력할 때 디렉토리인지 파일인지 여부를 판별하도록 하시오.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <dirent.h>
#include <stdio.h>
#include <fcntl.h>
#include <time.h>

char *rand_name(void)
{
	int i, len;
	static char buf[8] = "\0";

	memset(buf, 0, sizeof(buf));
	len = rand() % 4 + 4;

	for(i = 0; i < len; i++)
		buf[i] = rand() % 26 + 97;

	printf("buf = %s\n", buf);

	return buf;
}

void make_rand_dname(char **dname)
```

```
{
	int i;

	for(i = 0; i < 3; i++)
	{
		dname[i] = (char *)malloc(sizeof(char) * 8);
		strcpy(dname[i], rand_name());
	}
}

void make_rand_file(void)
{
	int i, j, fd, len, cnt;
	char buf[11] = "\0";

	len = rand() % 6 + 5;
	cnt = rand() % 4 + 2;

	for(i = 0; i < cnt; i++)
	{
		for(j = 0; j < len; j++)
			buf[j] = rand() % 26 + 97;

		fd = open(buf, O_CREAT, 0644);
		close(fd);

		memset(buf, 0, sizeof(buf));
	}
}

void lookup_dname(char **dname)
{
	int i;

	for(i = 0; i < 3; i++)
	{
		chdir(dname[i]);
		make_rand_file();
		chdir("..");
	}
}

void find_abc(void)
{
	struct dirent *p;
	struct stat buf;
	DIR *dp;
	int i, len;

	dp = opendir(".");

	while(p = readdir(dp))
	{
		stat(p->d_name, &buf);
		len = strlen(p->d_name);

		for(i = 0; i < len; i++)	// 파일명에 a,b,c 가 하나라도 존재하면 출력함
		{
			if(!strncmp(&p->d_name[i], "a", 1) ||
```

```
                !strncmp(&p->d_name[i], "b", 1) ||
                !strncmp(&p->d_name[i], "c", 1))
            {
                printf("name = %s\n", p->d_name);
                    break;
            }
        }
    }
    closedir(dp);
}

void recur_find(char **dn)
{
    int i;

    for(i = 0; i < 3; i++)
    {
        chdir(dn[i]);   // dn[i] 디렉토리를 현재 디렉토리로 함
        find_abc();
        chdir("..");    // 상위 디렉토리로 나옴
    }
}

int main(void)
{
    int i;
    char *dname[3] = {0};

    srand(time(NULL));
    make_rand_dname(dname);

    for(i = 0; i < 3; i++)
        mkdir(dname[i], 0755);

    lookup_dname(dname);
    recur_find(dname);

    return 0;
}
```