TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 하성용
accept0108@naver.com

20 일차
리눅스 프로그래밍 3 일차

quiz 2 답

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

typedef struct __queue
{
	int score;
	char *name;
	struct __queue *link;
} queue;

void disp_student_manager(int *score, char *name, int size)
{
	char *str1 = "학생 이름을 입력하시오: ";//모니터 상에 이름과 성적 출력
	char *str2 = "학생 성적을 입력하시오: ";
	char tmp[32] = {0}; //선언한 tmp

	write(1, str1, strlen(str1));
	read(0, name, size); //받아야되는 정보
	write(1, str2, strlen(str2));
	read(0, tmp, sizeof(tmp)); //캐릭터포인터자리에 int 형 포인터를 넣기위해 tmp 선언

	*score = atoi(tmp); //tmp 를 변환해서 캐릭터값 tmp 넣어줍니다
}

void confirm_info(char *name, int score)
{
	printf("학생 이름 = %s\n", name);
	printf("학생 성적 = %d\n", score);
}

queue *get_queue_node(void)
{
	queue *tmp;

	tmp = (queue *)malloc(sizeof(queue));
	tmp->name = NULL;
	tmp->link = NULL;

	return tmp;
}

void enqueue(queue **head, char *name, int score)
{
	if(*head == NULL)
	{
		int len = strlen(name);
		(*head) = get_queue_node();
		(*head)->score = score;
		(*head)->name = (char *)malloc(len + 1);
		strncpy((*head)->name, name, len);
		return;
	}
```

```
		enqueue(&(*head)->link, name, score);
}

void print_queue(queue *head)
{
		queue *tmp = head;

		while(tmp)
		{
			printf("name = %s, score = %d\n", tmp->name, tmp->score);
			tmp = tmp->link;
		}
}

void remove_enter(char *name)
{
		int i;

		for(i = 0; name[i]; i++)
			if(name[i] == '\n') //계행이면
				name[i] = '\0'; //null 문자로 대체
}//이름일땐 이렇게 되지만 문장이 될때는 계행이 될때의 문자값들을 다알아야함

int main(void)
{
		int cur_len, fd, btn = 0;
		int score; //두개의 변수를 전달해줘야함

		// Slab 할당자가 32 byte 를 관리하기 때문에 성능이 빠름
		char name[32] = {0}; //한국이름 길어도 5 자니 32 잡아줌
		char str_score[32] = {0};
		char buf[64] = {0};

		queue *head = NULL; //선언

		for(;;)//이 작업들이 반복되어야 저장이됨
		{
		printf("1 번: 성적 입력, 2 번: 파일 저장, 3 번: 파일 읽기, 4 번: 종료\n");//역할분할
		scanf("%d", &btn); //read 해서 swich 에 무언가 숫자값넣어주는게 보통
				//지금은 대신 scanf 로 btn 받기

		switch(btn) //어떤 키로 인해 동작하는 swich, btn 을 만듬
		{
			case 1:
/*인터페이스, 이름과 정보 받을수있도록 score, 이 작업들이 반복되어야함*/
			disp_student_manager(&score, name, sizeof(name));
				remove_enter(name);
				confirm_info(name, score);

				enqueue(&head, name, score);
				print_queue(head);
				break; //성적을 입력받고 큐에 다넣었으면 브레이크

		case 2:
		// 만약 파일 없다면 생성
		// 있다면 불러서 추가
		if((fd = open("score.txt", O_CREAT | O_EXCL | O_WRONLY, 0644)) < 0)
/*쓸수있는 권한이 없으니 WRONLY 를줌, 0 보다 작다면 즉,에러파일
  O_CREAT 를 같이 사용할때 O_EXCL 은 파일있으면 무시(보존)*/
```

```
		fd = open("score.txt", O_RDWR | O_APPEND);
		//APPEND 는 생성이 아니라서 0644 필요없음

			/* 어떤 형식으로 이름과 성적을 저장할 것인가 ?
			 저장 포맷: 이름,성적\n */
		strncpy(buf, name, strlen(name));
		/*버퍼에 네임이 복사가 되고 추가하려면 메인에 cur_len 값 추가
		cur_len = strlen(buf);
		//printf("cur_len = %d\n", cur_len);
		buf[cur_len] = ','; //뒤쪽연결위해 buf[cur_len]에 , 넣어줌
		sprintf(str_score, "%d", score);
		//버퍼값 하나받고 %d 로 바꿀형식 그리고 값에 해당하는 값해서 바꿔주는 형식의 sprintf
		strncpy(&buf[cur_len + 1], str_score, strlen(str_score));
/*시작하는 위치가 cur_len+1 이 되야하고 &으로 주소전달, 스코어가 지금은 숫자로 바뀌어있으니
문자로 바뀌워줘야해서 main 에 score 을 또 만들어줘야함,
이름이 중복되므로 str_score 만들어줌*/
		buf[strlen(buf)] = '\n'; //버퍼하고 버퍼를함 자기위치에 계행문자 삽입
		//printf("buf = %s, buf_len = %lu\n", buf, strlen(buf));

		write(fd, buf, strlen(buf)); //fd 에 write, 버퍼에있
					는 내용을 덮어씌워야함

			close(fd);

			break;

		case 3:
/*int fd 선언해서 open read 가능하게 만들어주고 2 번은 파일저장
if((fd = open("score.txt", O_RDONLY)) > 0) //괄호에 대해서 팁, if(fd=open()<0) 으로 만들
면 0 보다 작냐는뜻으로 결과가 0 혹은 1 이됨, 읽어서 집어넣을것
			{
			int i, backup = 0; //백업초기값 0,백업은 다복사되기때문에 조심
						// 이름 1,성적 1\n
						// 이름 2,성적 2\n
						// .....
						// 이름 n,성적 n\n
//이름 n,성적 n\n ->이렇게 다발로들어오면 어디서 끊기가 힘듬, 콤마가 오는 구간으로 구현이좋음
			read(fd, buf, sizeof(buf)); //fd 에서 읽은 값을 sizeof 버퍼만큼 읽는다

			for(i = 0; buf[i]; i++)
			{
				if(!(strncmp(&buf[i], ",", 1))) //콤마가 있는지확인, 여기서 걸
							리면 콤마가있는거
			{
			strncpy(name, &buf[backup], i – backup);
			//name 으로 받아들일 시점을 알기위해 버퍼를 받음 i 시점에서 걸림
				backup = i + 1;
			}
			if(!(strncmp(&buf[i], "\n", 1))) //계행문자 찾기
			{
		strncpy(str_score, &buf[backup], i – backup); //[backup]은 시작주소 같게
			backup = i + 1;
			enqueue(&head, name, atoi(str_score)); /*score 로 받았는데
str_score 로 바꿔주면 숫자값으로 다시 넘어감*/
			}
				}

			print_queue(head);
					}
			else //0 보다 크지않다면
```

```
                        break;

                        break;

                case 4:
                goto finish; //최적화하면서 문제생길수있느니 goto finish 해주기
                        break;

                        default:
                        printf("1, 2, 3, 4 중 하나 입력하셈\n");
                        break;
                }
        }

finish:
        return 0;
}

//치환명령어 :%s/OO/OO/g

세그멘테이션 폴트 오류떳을때

gdb.a.out
b main
r

p l
p name //네임값보기
p backup

strcmp//둘이같을땐 제로를 리턴한다



dup 이란?

dup1.c
#include<unistd.h>
#include<fcntl.h>
#include<stdio.h>

int main(void)
{
        int fd;
        fd=open("a.txt",O_WRONLY|O_CREAT|O_TRUNC,0644);
        close(1); //1 번이닫힘
        dup(fd); //dup 뜻: 복사하다 종료된걸 복사 fd 는 1 번의 역할을 하게됨
3 번인 a.txt 가 받아가게됨
        printf("출력될까?\n");
        return 0;
}

cat mytar.c > ccc
cat < ccc
cat ccc


 cat < ccc > ddd
```

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

int main(void)
{
        int fd;
        char buff[1024];
        fd=open("a.txt",O_RDONLY);
        close(0); //0 번(키보드) 닫힘 //한번닫으면 두번다시 못염
        dup(fd); //fd 가 0 번대체-파일자체가 0 번을대체 즉 입력자체가 파일이됨
        gets(buff); //문자열을 입력받을수있는 //이미 끝나서 gets 막히고 print 출력되고 끝남
        printf("출력될까?\n");
        return 0;
}
```

printf(buff) 추가

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

int main(void)
{
        int fd;
        char buff[1024];
        fd=open("a.txt",O_RDONLY);
        close(0);
        dup(fd);
        gets(buff); //gets 에서 fd 의 값을 받아서 출력
        printf("%s",buff);
        printf("출력될까?\n");
        return 0;
}
```

/*명령어는 Process 의 의 State 를 보는 명령어에 해당한다.

ps-ef|get bash|grep-v grep|awk ;ask '{point s$}'

pid 란 ex)주민번호, 즉고유값

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

int main(int argc, char *argv[])
{
        int i;
        char ch='a';
        int fd=open(argv[1],O_WRONLY|O_CREAT|O_TRUNC,0644);
        lseek(fd,512-1,SEEK_SET); //512 번째에 1 바이트를 쓰는데 a
        write(fd,&ch,1);
        close(fd);
        return 0;
}
```

명령어 주기

./a.out mbr.txt
xxd mbr.txt

mbr//마스터 부트 레코드
코드를 분석하게되면 보게되는데
Magic number

512 넘어가는구간
메모리 다잡고가는것

Principle of Multi-Tasking*

fifo.c

수행하고 다른 terminal 에서 cat > myfifo 를 수행함

블로킹이란?

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

int main(void)
{
	int fd, ret;
	char buf[1024];
	mkfifo("myfifo");
	fd=open("myfifo",O_RDWR); //마이피포열어서 디스크립터줌
	for(;;)
	{
		ret=read(0,buf,sizeof(buf)); //0 번 읽고 키보드입력을 받겟다~
블로킹이라 키보드입력이 들어올때까지는 기다리고있다 키보드입력은 버퍼에 저장
		buf[ret-1]=0;
		printf("Keyboard Input:[%s]\n",buf);
		read(fd,buf,sizeof(buf)); //리드라는 함수는 블로킹함수 //비효율적
		buf[ret-1]=0;
		printf("Pipe Input:[%s]\n",buf);
	}
	return 0;
}
```

터미널창에서 mkfifo myfifo 치고
ls
하면 myfifo 노랑색이 생성되있을거
위에 컴파일프로그램을 실행

후에 커널을 하나더켜서 같은위치에서
cat > myfifo 실행하면 자신과 통신을 할수있게됨

ls -al /dev
하면 여러가지가 보임

앞에 아무것도 없는건 그냥파일
p 는 파이프
파이프는 로컬상에 통신으로 사용됨

ex)sp -ef | grap bash

d 는 디렉토리
c 는 캐릭터디바이스
b 는 블록디바이스

블록은 어떤 특정한단위가있는거
어떤특정한단위로 움직임
메모리의 최소단위 2 의 12 승 바이트로 화산하면 4kb = 페이지
하드디스크도 4k 방식으로 동작
즉 하드디스크는 블록, 디렘 블록
렘은 렘덤 엑세스 메모리 = 순서지키지말라는뜻

캐릭터는 순서가있음
순서가 지켜져야함(키보드, 모니터
ex)hello 를 보내야할곳에 error 메시지를 보내고 error 메시지를 보내야할장소에 hello 를 보내는것

논 블로킹이란?

non_blocking1.c

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

int main(void)
{
	int fd, ret;
	char buf[1024];
	fd = open("myfifo",O_RDWR);
	fcntl(0,F_SETFL,O_NONBLOCK); //파일에 무언가 권한을 세팅해주는것(뒤쪽에있는 논블럭으로 세팅 ) 읽을게 없으면 넘기고 읽을게 있으면 보는게 논블럭, 즉 입력을 논블럭으로 설정하겟다는거
	fcntl(fd,F_SETFL,O_NONBLOCK);//0
	for(;;)
	{
		if((ret=read(0,buf,sizeof(buf)))>0)
		{
			buf[ret-1]=0;
			printf("Keyboard Input:[%s]\n",buf);
		}
		if((ret=read(fd,buf,sizeof(buf)))>0)
		{
			buf[ret-1]=0;
			printf("Pipe Input:[%s]\n",buf);
		}
	}
	close(fd);
	return 0;
}
```