TI DSP, MCU, Xilinx Zynq FPGA Based Programming Expert Program

Instructor – Innova Lee(Sanghoon Lee)
gcccompil3r@gmail.com
Student – Hyungju Kim
mihaelkel@naver.com

```
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <math.h>
#define ERROR -1
typedef struct __student{
	char name[20];
	int score;
}student;

typedef struct __queue{
	student st;
	int data;
	struct __queue* link;
}queue;

queue* get_node(void){
	queue* tmp;
	tmp = (queue*)malloc(sizeof(queue)*1);
	tmp->link = NULL;
	return tmp;
}

void ins_queue(queue** head,student st){
	queue** tmp = head;
	while(*tmp)
		tmp = &(*tmp)->link;
	*tmp = get_node();
	(*tmp)->st = st;
}

void del_queue(queue** head,int index){
	int i;
	queue* tmp = *head;
	queue* parent = NULL;
	if(index == 1){
		(*head) = (*head)->link;
		free(tmp);
		return;
	}
	for(i=0;i<index-1;i++){
		parent = tmp;
		tmp = tmp->link;
	}
	parent->link = tmp->link;
	free(tmp);
}
```

```
void print_queue(queue** head){
	queue* tmp = *head;
	student p;
	int index = 1;
	char buf[1024]={'\0'};
	char title[1024]={'\0',};
	sprintf(title,"%5s\t%8s\t%4s\n","index","name","score");
	write(1,title, strlen(title));

	while(tmp){
		p = tmp->st;
		sprintf(buf,"%5d\t%8s\t%4d\n",index++,p.name,p.score);

		write(1,buf, strlen(buf));
		tmp = tmp->link;
	}
}

void student_to_str(student p,char* buf){

	char* name = p.name;
	int score = p.score;

	sprintf(buf,"%8s\t%4d\n", name, score);
}

void init_db(int fd){
	char title[1024]={'\0',};
	sprintf(title,"%5s\t%8s\t%4s\n","index","name","score");
	write(1,title, strlen(title));
	write(fd,title, strlen(title));
}

void save_file(queue** head,int fd,char* path){
	queue* tmp = *head;
	char buf[1024] = {'\0'};
	student p;
	int index = 1;
	char title[1024]={'\0',};

	sprintf(title,"%5s\t%8s\t%4s\n","index","name","score");
	//reopen to add O_TRUNC option.
	fd = open(path,O_RDWR | O_TRUNC,0644);
	write(fd,title, strlen(title));
	while(tmp){
		p = tmp->st;
		sprintf(buf,"%5d\t%8s\t%4d\n",index++,p.name,p.score);
		write(fd, buf , strlen(buf));
```

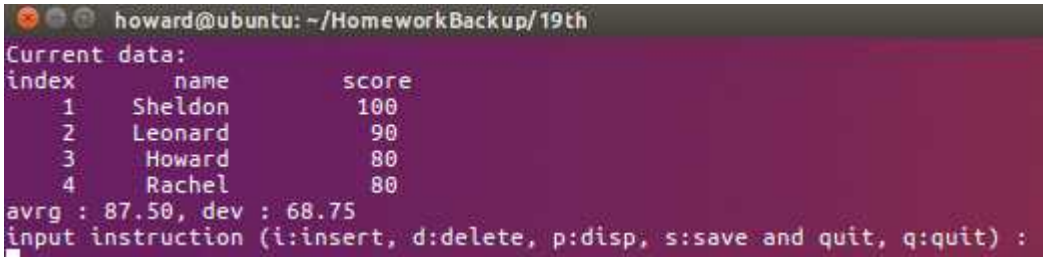
```
		tmp = tmp->link;
	}
	close(fd);
}

void statistics(queue** head){
	queue* tmp = *head;
	student p;
	int index = 0,sum = 0, i = 0;
	int t[100]={0,};
	double avrg = 0, dev = 0;
	while(tmp){
		p = tmp->st;
		t[index] = p.score;
		sum += t[index];
		index++;
		tmp = tmp->link;
	}
	avrg = (double)sum/index;
	for(i=0;t[i];i++){
		dev += (t[i] - avrg)*(t[i] - avrg);
	}
	dev = dev / index;
	printf("avrg : %.2lf, dev : %.2lf\n",avrg,dev);
}



void start_db(queue** head, int fd,char* path){
	queue* tmp = *head;
	student st;
	student st_init = {"",0};
	student p;
	char buf[1024]={'\0',};
	char ch[2], ins[2];
	int index = 0;
	int i = 0;

	while(1){
		sprintf(buf,"input instruction (i:insert, d:delete, p:disp, s:save and quit, q:quit) :\n");
		write(1,buf,strlen(buf));
		read(0,ch,2);
		system("clear");
		switch(ch[0]){		//ch[1] has '\n'
			case 'q':
				break;
			case 'p':
				printf("Current data: \n");
				print_queue(head);
```

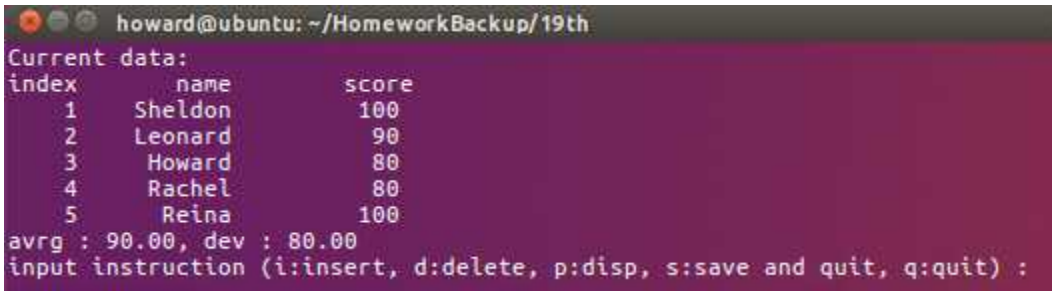
```
                statistics(head);
                break;
            case 'i':
                //st receive
                st = st_init;

                sprintf(buf,"input name : ");
                write(1, buf, strlen(buf));
                read(0, st.name, sizeof(st.name));

                //'\n' elliminate
                while(st.name[i] != 10)
                    i++;
                st.name[i] = 0;
                i=0;

                sprintf(buf,"input score : ");
                write(1, buf, strlen(buf));

                read(0, buf, sizeof(buf));
                st.score = atoi(buf);

                ins_queue(head,st);
                break;
            case 's':
                printf("database saved!\n");
                save_file(head,fd,path);
                break;
            case 'd':
                //deleted index receive
                sprintf(buf,"input delete index : ");
                write(1, buf, strlen(buf));
                read(0,ins,sizeof(ins));

                index = atoi(ins);
                del_queue(head,index);
                break;
            default :
                break;
        }
        if(ch[0] == 's'|| ch[0] == 'q')
            break;
    }
    sprintf(buf,"Database Quit\n");
    write(1, buf, strlen(buf));

}

void open_db_file(queue** head,int* fd,char* path){
```

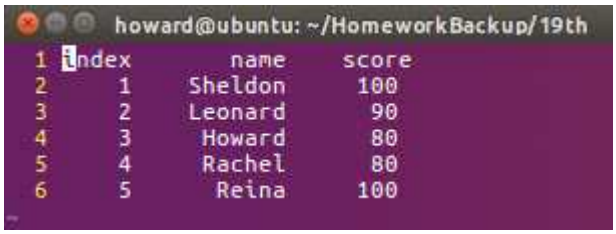
```
	(*fd) = open(path,O_RDWR,0644);

	if((*fd) == ERROR){
		(*fd) = creat(path,0644);
		return;
	}
	char ch;
	char name[100] = {'\0',};
	int score = 0;
	int index = 1, i = 0, flag = 0;
	int len;
	student st;
#if 0
	//파일 읽어오기. 미완성
	//'\t'으로 문자 구분, '\n'으로 data 구분
	//student 구조체 변수의 name, score에 저장.
	//구조체 변수를 ins_queue
	//flag 0 : index 무시
	//flag 1 : name
	//flag 2 : score
	while(read(*fd,&ch,1)){
		if(ch == '\n'){
			index++;
			i = 0;
		}
		if(ch != '\n' && ch != '\t' && ch != ' '){
			if(flag == 1)
				name[i++] = ch;

			else if (flag == 2)
				score = ch+48;

			else
				flag = 1;
		}
		else if(ch == '\t'){
			if(flag == 1)
				flag = 2;
			else if(flag == 2)
				flag = 0;
		}
	}
#endif
}

int main(int argc,char* argv[]){
	int fd;
	queue* head = NULL;
```

```
        //open saved database file
        open_db_file(&head,&fd,argv[1]);

        //start database
        start_db(&head,fd,argv[1]);
        close(fd);
}
```

Result :







Haven't yet Implemented :

1.sort (by descent and ascent)

2.load text file when the program starts.