Design your chatting app GUI including basic input and output functions.

The goal of homework 3 is to design and implement the chatting app with basic input and out functions.

Logging in Accounts

OOutput

Current time

Elapsed time

Oinput

[Guide line]

No need to use fork(), thread synchronization, and etc.

Modify “chat.c” I showed this week as the follwoings:

[1] Use ncurses to divide the main screen into 4 sub-windows

[2] Each sub-windows has box line.

* Terminal size is 80 X 24
* Output window size is 60 X 20
* Input window size is 60 X 4
* “Logging in Accounts” window size is 20 X 20
* “Time” window size is 20 X 4

[3] Display the current time and elapsed time (time will be refreshed in every 500ms)

[4] When a person logs in, his/her account will be appeared in the “Logging in Accounts” window.

When he/she type “/bye”, his/her account will be disappeared in the “Logging in Accounts” window.

(Question: do we need another shared memory?)

[5] Try to do your best to apply your output produced in Homework#2 to this homework!!

[6] How to test your app:

* Open the first terminal and run chat Jico to write text messages (Actually, you don’t need to create a new user account “Jico”. It is enough to use the terminal where you logged in)
* Open the second terminal and run chat Izzy to write text messages
* Open the third terminal and run chat GD to write text messages
* Chat sequences are randomly and one person by one person. We do not consider that three persons chat at the same time. (We need threads. However, we do not need thread synchronization, fork, and etc for this work.)

**=====================================================================**

**1. Put your program source as here (Do not put the screen shot of your source code!! If you insist, you will get zero point)**

**1) makefile**

**CC=gcc**

**CFLAGES=-g -Wall**

**all :chat**

**chat: chat.o**

**$(CC) $(CFLAGES) -o $@ $^ -lncurses -lpthread**

**chat.o: chat.c**

**clean:**

**rm -f \*.o**

**rm -f chat**

**2) chat.c**

**/\***

**\* chat.c**

**\* Hw#3 Make Simple Chat Ncurses**

**\* Lee Jae Yoon(이재윤), 201624548**

**\* 유닉스 응용 프로그래밍(CP33357-059)**

**\*/**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <unistd.h>**

**#include <pthread.h>**

**#include <ncurses.h>**

**#include <sys/ipc.h>**

**#include <sys/shm.h>**

**#define BUFFSIZE 1024**

**#define MAXACCOUNT 20**

**#define NAMESIZE 20**

**typedef struct message\_buffer {**

**char name[NAMESIZE];**

**char msg[BUFFSIZE];**

**char account[MAXACCOUNT][NAMESIZE];**

**int id;**

**} MSG\_BUFF;**

**MSG\_BUFF \* msg\_buff;**

**int is\_running;**

**int account\_cnt;**

**void \*print\_chat();**

**void \*get\_input();**

**void \*recv\_send();**

**void chat();**

**void cleanup();**

**void die(char \* msg);**

**void setbox();**

**void refreshing();**

**void \*show\_time();**

**void \*show\_account();**

**int shmid;**

**char userID[NAMESIZE];**

**void \* shmaddr = (void\*) 0;**

**WINDOW \*terminal\_scr;**

**WINDOW \*output\_scr;**

**WINDOW \*input\_scr;**

**WINDOW \*account\_scr;**

**WINDOW \*time\_scr;**

**int main(int argc, char\* argv[])**

**{**

**if (argc < 2) {**

**fprintf(stderr, "[Useage]: ./chat UserID \n");**

**exit(-1);**

**}**

**memset(userID , 0, NAMESIZE);**

**strcpy(userID, (const char\*)argv[1]);**

**shmid = shmget((key\_t)20200406, sizeof(MSG\_BUFF), 0666|IPC\_CREAT);**

**shmaddr = shmat(shmid, (void \*)0, 0666);**

**msg\_buff = (MSG\_BUFF \*) shmaddr;**

**for (int i = 0; i < MAXACCOUNT; i++) {**

**if (msg\_buff->account[i][0] == '\0') {**

**strcpy(msg\_buff->account[i], (const char\*)argv[1]);**

**account\_cnt = i;**

**break;**

**}**

**}**

**initscr();**

**chat();**

**endwin();**

**return 0;**

**}**

**void chat()**

**{**

**curs\_set(0);**

**scrollok(input\_scr, TRUE);**

**terminal\_scr = newwin(24, 80, 0, 0);**

**output\_scr = subwin(terminal\_scr, 20, 60, 0, 0);**

**input\_scr = subwin(terminal\_scr, 4, 60, 20, 0);**

**account\_scr = subwin(terminal\_scr, 20, 20, 0, 60);**

**time\_scr = subwin(terminal\_scr, 4, 20, 20, 60);**

**mvwhline(output\_scr, 1, 1, 68, 0);**

**wprintw(output\_scr, " \*\*\*\*\* Type /exit to quit !! \*\*\*\*\* \n\n");**

**setbox();**

**refreshing();**

**is\_running = 1;**

**pthread\_t thread[4];**

**pthread\_create(&thread[0], NULL, get\_input, NULL);**

**pthread\_create(&thread[1], NULL, recv\_send, NULL);**

**pthread\_create(&thread[2], NULL, show\_account, NULL);**

**pthread\_create(&thread[3], NULL, show\_time, NULL);**

**pthread\_join(thread[0], NULL);**

**pthread\_join(thread[1], NULL);**

**pthread\_join(thread[2], NULL);**

**pthread\_join(thread[3], NULL);**

**delwin(terminal\_scr);**

**delwin(output\_scr);**

**delwin(input\_scr);**

**delwin(account\_scr);**

**delwin(time\_scr);**

**}**

**void \*show\_account()**

**{**

**int cnt;**

**while (is\_running) {**

**cnt = 1;**

**werase(account\_scr);**

**for (int i = 0; i < MAXACCOUNT; i++) {**

**if (msg\_buff->account[i][0] != '\0') {**

**mvwprintw(account\_scr, cnt++, 2,"%s", msg\_buff->account[i]);**

**}**

**}**

**box(account\_scr, ACS\_VLINE, ACS\_HLINE);**

**wrefresh(account\_scr);**

**usleep(300);**

**}**

**}**

**void \*show\_time()**

**{**

**time\_t start,now;**

**start = time(0);**

**struct tm ts;**

**char buf[80];**

**while(is\_running){**

**//시간 출력**

**time(&now);**

**ts = \*localtime(&now);**

**strftime(buf, sizeof(buf), "%H-%M-%S",&ts);**

**mvwprintw(time\_scr, 1,2,"%s",buf);**

**now = time(0);**

**time\_t tmp = now-start;**

**ts = \*localtime(&tmp);**

**// Match with our time**

**ts.tm\_hour -= 9;**

**strftime(buf, sizeof(buf), "%H-%M-%S",&ts);**

**mvwprintw(time\_scr, 2,2,"%s",buf);**

**box(time\_scr, ACS\_VLINE, ACS\_HLINE);**

**wrefresh(time\_scr);**

**usleep(500);**

**}**

**}**

**void \*get\_input()**

**{**

**char tmp[BUFFSIZE];**

**int send\_id = 1;**

**while (is\_running) {**

**mvwgetstr(input\_scr, 1, 1, tmp);**

**sprintf(msg\_buff->msg, " %s\n", tmp);**

**strcpy(msg\_buff->name, userID);**

**msg\_buff->id = send\_id;**

**if (strcmp (msg\_buff->msg, " /exit\n") == 0 ) {**

**die("exit");**

**}**

**wprintw(output\_scr, " [Send : %d] > %s", send\_id++, msg\_buff->msg);**

**werase(input\_scr);**

**setbox();**

**refreshing();**

**usleep(100);**

**}**

**}**

**void setbox() {**

**box(output\_scr, ACS\_VLINE, ACS\_HLINE);**

**box(input\_scr, ACS\_VLINE, ACS\_HLINE);**

**box(account\_scr, ACS\_VLINE, ACS\_HLINE);**

**box(time\_scr, ACS\_VLINE, ACS\_HLINE);**

**wrefresh(input\_scr);**

**}**

**void refreshing() {**

**wrefresh(output\_scr);**

**wrefresh(input\_scr);**

**wrefresh(account\_scr);**

**wrefresh(time\_scr);**

**}**

**void \*recv\_send()**

**{**

**MSG\_BUFF bef\_msg;**

**strcpy(bef\_msg.msg, msg\_buff->msg);**

**strcpy(bef\_msg.name, msg\_buff->name);**

**bef\_msg.id = msg\_buff->id;**

**int myrecv = 1;**

**while (is\_running) {**

**if (strcmp(msg\_buff->msg, " .exit\n") == 0) {**

**fprintf(stderr, "Chat is closed\n");**

**is\_running = 0;**

**}**

**else {**

**if ( ( strcmp(msg\_buff->msg, bef\_msg.msg) != 0 ||**

**strcmp(msg\_buff->name, bef\_msg.name) != 0 ||**

**bef\_msg.id != msg\_buff->id ) &&**

**strcmp(msg\_buff->name,userID) != 0 ) {**

**if (strcmp(msg\_buff->msg, " /exit\n") == 0) {**

**wprintw(output\_scr, " [Exit %s] Goodbye %s !\n",**

**msg\_buff->name, msg\_buff->name);**

**strcpy(bef\_msg.msg, msg\_buff->msg);**

**}**

**else {**

**wprintw(output\_scr, " [Recv #%d By %s] > %s",**

**myrecv++, msg\_buff->name, msg\_buff->msg);**

**}**

**bef\_msg.id = msg\_buff->id;**

**strcpy(bef\_msg.msg, msg\_buff->msg);**

**strcpy(bef\_msg.name, msg\_buff->name);**

**box(output\_scr, ACS\_VLINE, ACS\_HLINE);**

**wrefresh(output\_scr);**

**wrefresh(input\_scr);**

**}**

**}**

**}**

**}**

**void cleanup()**

**{**

**memset(msg\_buff->account[account\_cnt] , 0, NAMESIZE);**

**delwin(terminal\_scr);**

**delwin(output\_scr);**

**delwin(input\_scr);**

**delwin(account\_scr);**

**delwin(time\_scr);**

**endwin();**

**shmdt(shmaddr);**

**}**

**void die(char \* msg)**

**{**

**cleanup();**

**perror(msg);**

**exit(-1);**

**}**

**3) chat.c에 대한 idea 설명**

**// 이름과 메시지, Account와 메시지 id를 공유하는 버퍼**

**typedef struct message\_buffer {**

**char name[NAMESIZE];**

**char msg[BUFFSIZE];**

**char account[MAXACCOUNT][NAMESIZE];**

**int id;**

**} MSG\_BUFF;**

**메시지, 이름, 메시지 id와 계정을 공유하는 구조체를 선언하여, 계정 리스트 출력과 중복 방지를 하였다.**

**계정 리스트 출력은 공유메모리를 제외한 파일입출력 시스템을 고려해 보았으나, 공유메모리를 사용하는 김에 공유메모리를 사용하였다.**

**// recv\_send 에 사용되는 함수인데 메시지 내용이 이전과 다르거나, 보낸이의 이름이 다르거나, 메세지 아이디가 다른 상태에서**

**// 보낸 이가 내가 아니라면 출력을 하는 시스템이다. 고민하는데 꽤나 고생했다.**

**if ( ( strcmp(msg\_buff->msg, bef\_msg.msg) != 0 ||**

**strcmp(msg\_buff->name, bef\_msg.name) != 0 ||**

**bef\_msg.id != msg\_buff->id ) &&**

**strcmp(msg\_buff->name,userID) != 0 )**

**// 내 아이디의 위치와, 메모리 정리를 위한account number을 선언**

**int account\_cnt;**

**4) chat.c에 포함된 함수 설명**

**/\***

**// MAXACCOUNT 만큼 루프를 돌며, NULL이 아닌 객체를 출력.**

**// 모든 프로세스는 시작 시 msg\_buff->account[account\_cnt]에 자기 아이디를 복사**

**\*/**

**void \*show\_account();**

**// 시간 출력 HW #1을 기용하였음. usleep을 이용해 0.5초 쉬는 시간을 주었음.**

**void \*show\_time();**

**/\***

**// 입력받아 버퍼에 저장한다.**

**// id에는 send\_id를 저장하는데, 항상 자기를 기준으로 하므로, 자신이 보낸 메시지는 id가 겹칠일이 절대 없다.**

**\*/**

**void \*get\_input();**

**/\***

**// 받은 메시지 버퍼의 내용을 출력한다.**

**// myrecv는 내가 받은 내용의 카운트를 출력한다. 나를 기준으로 출력해야 하기 때문이다.**

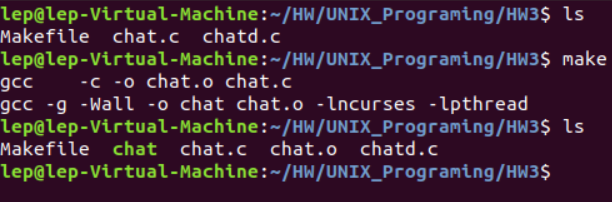
**// idea설명에서 말했던 if문을 사용하여 출력하므로, 똑같은 내용을 출력하거나, 받은 내용을 빼 먹을 일이 없다.**

**\*/**

**void \*recv\_send()**

**2. You must show the building result after compiling and linking your source codes. You must show no warnings and errors (Use gcc -Wall option).**

**(Put a screen shot of your C debugging output)**

****

**3. Put a screen shot of output generated by your program. Your output screen shot must be readable for me to verify your chat program. (쓰레드 동기화를 이용하지 말라고 하여, 쓰레기 값이 보일 수 있습니다.)**

