**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)**

chat.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <pthread.h>

#include <ncurses.h>

#include <time.h>

#include <sys/shm.h>

*// define buffer size fo saving message*

#define BUFFSIZE 1024

*// Declare Windows*

WINDOW \*base\_scr, \*input\_scr, \*output\_scr, \*account\_scr, \*time\_scr;

*// Define struct for saving input or output data*

**struct** message\_buffer {

**char** name[20];

**char** msg[BUFFSIZE];

**int** id;

};

*// Define struct for saving user information*

**typedef** **struct** checkInfo {

**char** userID[20];

**char** message[40];

**char** messageTime[20];

**int** messageID;

*// Check the number of user in chat room*

**int** user\_count;

*// Variable for saving user name*

**char** log[10][20];

} CHAT\_INFO;

*// Declare two instances of message\_buffer struct for saving in & out message*

**struct** message\_buffer buff\_in;

**struct** message\_buffer buff\_out;

*// Declare for getting time*

time\_t now;

**struct** tm t;

*// Declare Flag variable that indicates state now*

**int** is\_running;

*// Declare chat information in global variable*

CHAT\_INFO \*chatInfo;

*// Declare global variable for saving time now*

**char** time\_now[50];

*// Define functions*

**void** print\_chat();

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

**void** chat();

**void** cleanup();

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

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

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

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

*// Declare for saving shared memory id*

**int** shmID;

*// Declare for saving user name*

**char** userID[20];

*// Allocate memory 10 times of checkInfo struct to chatInfo*

chatInfo = (CHAT\_INFO \*)malloc(**sizeof**(**struct** checkInfo) \* 10);

*// Type casting from int to void\**

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

*// if user doesn't type name, the error message appears*

**if**(argc < 2){

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

exit(-1);

}

*// Saving user name in userID variable*

strcpy(userID, argv[1]);

*// Create shared memory id which key is 20200421 exclusively*

shmID = shmget((key\_t)20200421, **sizeof**(CHAT\_INFO), 0666|IPC\_CREAT|IPC\_EXCL);

*// If there's already shared memory exists, don't create and just get shared memory id*

**if**(shmID < 0){

shmID = shmget((key\_t)20200421, **sizeof**(CHAT\_INFO), 0666);

}

*// Set shared memory address*

shmaddr = shmat(shmID, (**void** \*)0, 0000);

*// Set shared memory address in chatInfo*

chatInfo = (CHAT\_INFO \*) shmaddr;

*// saving user name in account list, buffer in name*

strcpy(chatInfo->log[chatInfo->user\_count], userID);

strcpy(buff\_in.name, userID);

*// Increase user count that indicates the number of memebers in chat room*

chatInfo->user\_count++;

*// Initiating screen*

initscr();

*// Call chat function*

chat();

**return** 0;

}

**void** chat(){

*// Set windows' size*

base\_scr = newwin(24, 80, 0, 0);

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

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

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

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

*// Set box line in subwindows*

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);

*// Refresh subwindows*

wrefresh(output\_scr);

wrefresh(input\_scr);

wrefresh(account\_scr);

wrefresh(time\_scr);

*// Set scroll ok*

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

*// Print message on output screen and draw box line, refresh*

mvwprintw(output\_scr, 0, 1, "\n \*\*\*\*\* Type /bye to quit!! \*\*\*\*\* \n\n");

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

wrefresh(output\_scr);

*// Initialize buffer ids and flag*

buff\_in.id = 0;

buff\_out.id = 0;

is\_running = 1;

*// Declare thread for controlling chat program*

pthread\_t thread[3];

*// Create 3 threads*

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

pthread\_create(&thread[1], **NULL**, show\_time, **NULL**);

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

*// Wait until all thread is finished*

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

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

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

*// Delete all windows*

delwin(base\_scr);

delwin(input\_scr);

delwin(output\_scr);

delwin(account\_scr);

delwin(time\_scr);

}

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

*// Declare buffer for saving input message*

**char** tmp[BUFFSIZE];

*// Running until flag is changed*

**while**(is\_running){

*// Get string from input screen to buffer*

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

*// Saving buffer's message to buff\_in.msg*

sprintf(buff\_in.msg, "%s\n", tmp);

*// if user type /bye*

*// Change account list and refresh account screen*

**if**(strcmp(buff\_in.msg, "/bye\n") == 0) {

**int** i = 0;

**int** j = 0;

**for**(i=0; i<chatInfo->user\_count; i++){

**if**(strcmp(chatInfo->log[i], buff\_in.name) == 0){

**for**(j=i+1; j<chatInfo->user\_count; j++){

strcpy(chatInfo->log[j-1], chatInfo->log[j]);

strcpy(chatInfo->log[j], "");

}

}

}

*// Decrease user count*

chatInfo->user\_count = chatInfo->user\_count - 1;

werase(account\_scr);

wrefresh(account\_scr);

*// Exit*

die("exit");

}

*// If user type ..*

*// Call print\_chat function*

**else** **if**(strcmp(buff\_in.msg, "..\n") == 0){

print\_chat();

}

*// If user type any message except last two*

*// Saving user information and time in chatInfo list*

*// messageID is unique for each message*

**else**{

strcpy(chatInfo[chatInfo->messageID].userID, buff\_in.name);

strcpy(chatInfo[chatInfo->messageID].messageTime, time\_now);

strcpy(chatInfo[chatInfo->messageID].message, buff\_in.msg);

*// Increase messageID*

chatInfo->messageID++;

*// Print user message on output screen*

    wprintw(output\_scr, " [Send : %s] > %s", buff\_in.name, buff\_in.msg);

*// Increase buffer id*

buff\_in.id++;

}

*// Refresh output screen and input screen*

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

wrefresh(output\_scr);

werase(input\_scr);

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

wrefresh(input\_scr);

usleep(100);

}

**return** 0;

}

**void** print\_chat(){

**int** i=0;

*// If user type ..*

*// Print all of messages in shared memory except user's own messages*

**for**(i=0; i<chatInfo->messageID; i++){

memset(&\*buff\_out.msg, 0, BUFFSIZE);

sprintf(buff\_out.msg, " [Recv : %s [%s]] %s",

chatInfo[i].userID, chatInfo[i].messageTime, chatInfo[i].message);

**if**(strcmp(chatInfo[i].userID,  buff\_in.name) != 0){

wprintw(output\_scr, buff\_out.msg);

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

wrefresh(output\_scr);

}

}

usleep(30000);

}

*// Delete all windows*

**void** cleanup(){

delwin(base\_scr);

delwin(input\_scr);

delwin(output\_scr);

delwin(account\_scr);

delwin(time\_scr);

endwin();

}

*// Delete all windows and exit*

**void** die(**char** \*s){

delwin(input\_scr);

delwin(output\_scr);

delwin(account\_scr);

delwin(time\_scr);

endwin();

perror(s);

exit(1);

}

*// This function shows current time and elapsed time*

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

**int** hour, min, sec;

**int** elapsed = 0;

**int** count = 0;

*// Running until flag is changed*

**while**(is\_running){

*// Get current time*

time(&now);

t = \*localtime(&now);

*// This function sleeps in 500ms so we have to set count 2*

*// because 1s = 500ms*

**if**(count == 2){

elapsed++;

*// Change elapsed time to time form*

sec = elapsed % 60;

min = (elapsed / 60) % 60;

hour = elapsed / 3600;

*// set count 0*

count = 0;

}

*// Saving current time in form that we want*

strftime(time\_now, **sizeof**(time\_now), "%H:%M:%S", &t);

*// Print current time and elapsed time on time screen and refresh*

mvwprintw(time\_scr, 1, 1, time\_now);

mvwprintw(time\_scr, 2, 1, "%02d:%02d:%02d", hour, min, sec);

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

wrefresh(time\_scr);

usleep(500000);

*// Increase count in 500ms*

count = count + 1;

}

**return** 0;

}

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

*// Running until the flag is changed*

**while**(is\_running){

**int** i=0;

*// Erase account list for showing account list's any changes*

werase(account\_scr);

*// Print account list on account screen using user count*

**for**(i=0; i<chatInfo->user\_count; i++){

mvwprintw(account\_scr, i+1, 1, chatInfo->log[i]);

}

*// Refresh account screen*

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

wrefresh(account\_scr);

sleep(1);

}

**return** 0;

}

**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.**

****