Name : Dev Adnani  
SID : 202212012

Subject : Systems Programming

Assignment : 8

1:

202212012\_Lab8\_1.c

#include <stdio.h>

#include <unistd.h>

#include <signal.h>

#include <stdlib.h>

void alarmhandler() {

printf("Alert! Too much time to enter string!\n");

exit(-1);

}

int main(void) {

alarm(10);

signal(SIGALRM,alarmhandler);

char input[100];

if(scanf("%s",input) ) {

printf("Input entered within 10 seconds.\n");

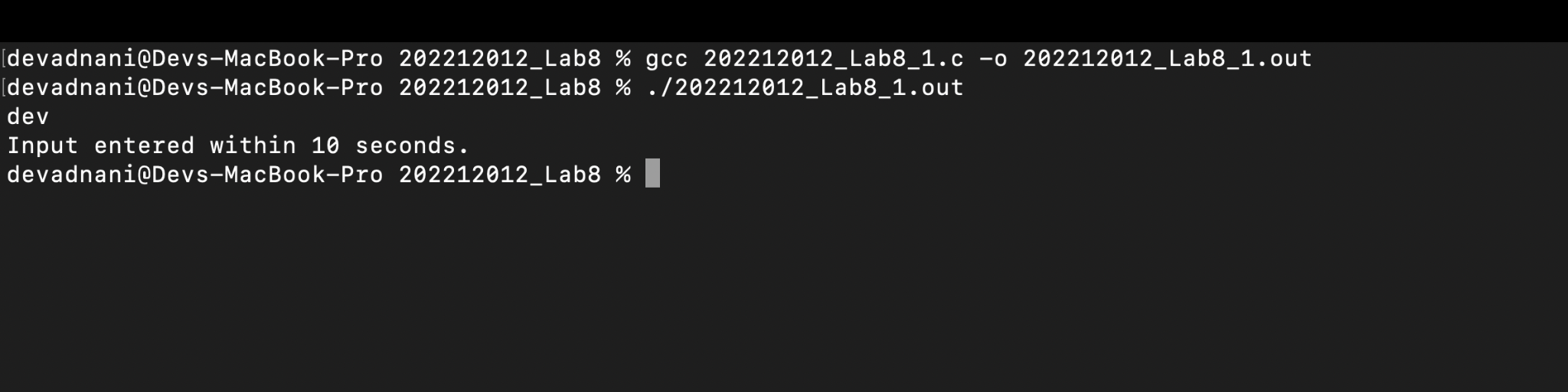
alarm(0);

exit(0);

}

}

Screenshot :



2: 202212012\_Lab8\_2.c

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <limits.h>

#include <string.h>

#include <sys/stat.h>

#include <dirent.h>

#include <fcntl.h>

#include <libgen.h>

#define \_POSIX\_SOURCE

#include <sys/types.h>

#include <signal.h>

#include <unistd.h>

#include <sys/wait.h>

#define READ 0

#define WRITE 1

int fd1[2], fd2[2], bytesread;

char message[100];

int i, pid1, pid2, status;

int que;

char \*questions[] = {"quit", "Yo Homie,Whats Your Name?", "which course are you hate the most?","what is your area of interest?"};

char \*answers[] = {"quit", "Dev", "Systems Programming & Computer Networks", "App/Backend Development"};

static void signalhandler(int signal) {

switch (signal) {

case SIGUSR1:

printf("Process %d: received SIGUSR1 \n", getpid());

if (pid1 == getpid()) {

} else {

bytesread = read(fd1[READ], message, 100);

printf("Child: Read %d bytes: %s \n", bytesread, message);

int ind = -1;

for (int i = 0; i <= 3; i++) {

if (strcmp(message, questions[i]) == 0) {

ind = i;

break;

}

}

printf("Child: Sending message \'%s\' to parent\n", answers[ind]);

write(fd2[WRITE], answers[ind], strlen(answers[ind]) + 1); /\* Send \*/

printf("Process %d is passing SIGUSR2 to %d...\n", getpid(), pid1);

kill(pid1, SIGUSR2);

if (ind == 0) {

close(fd1[READ]);

close(fd2[WRITE]);

exit(0);

}

}

break;

case SIGUSR2:

printf("Process %d: received SIGUSR2 \n", getpid());

if (pid1 == getpid()) {

bytesread = read(fd2[READ], message, 100);

printf("Parent: Read %d bytes: %s \n", bytesread, message);

if (que == 0) {

close(fd1[WRITE]);

close(fd2[READ]);

wait(&status);

exit(0);

}

printf("PLease enter q. number: ");

scanf("%d", &que);

close(fd1[READ]);

close(fd2[WRITE]);

printf("Parent: Sending message \'%s\' to child\n", questions[que]);

write(fd1[WRITE], questions[que], strlen(questions[que]) + 1);

kill(pid2, SIGUSR1);

}

break;

default:

break;

}

return;

}

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

pipe(fd1);

pipe(fd2);

if (signal(SIGUSR1, signalhandler) == SIG\_ERR)

printf("Pærent: Unable to create handler for SIGUSR1\n");

if (signal(SIGUSR2, signalhandler) == SIG\_ERR) {

printf("Pærent: Unable to create handler for SIGUSR2\n");

}

pid1 = getpid();

if ((pid2 = fork()) == 0) {

while (1) {

pause();

}

} else {

printf("PLease enter q. number: ");

scanf("%d", &que);

close(fd1[READ]);

close(fd2[WRITE]);

printf("Parent: Sending message \'%s\' to child\n", questions[que]);

write(fd1[WRITE], questions[que], strlen(questions[que]) + 1);

kill(pid2, SIGUSR1);

while (1) {

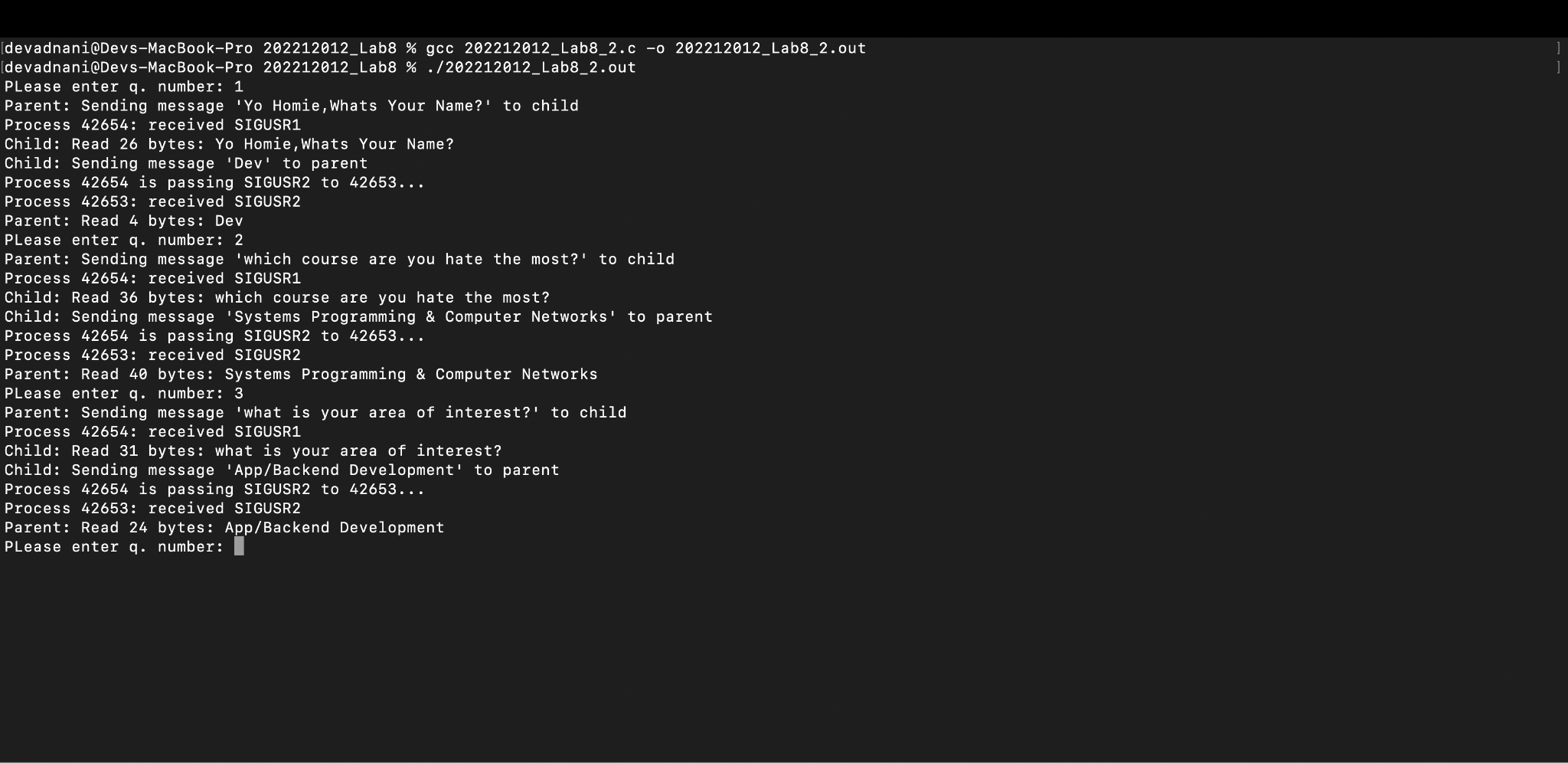
pause();

}

}

}

Output :



3:

202212012\_Lab8\_3\_sendsignal.c // This Is A Common File For All Three Programs

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <limits.h>

#include <string.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <unistd.h>

#include <dirent.h>

#include <fcntl.h>

#include <libgen.h>

#define \_POSIX\_SOURCE

#include <sys/types.h>

#include <signal.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

printf("Enter process id: ");

int pid, id;

scanf("%d", &pid);

printf("Press 1, if you want to send the signal to the process or \nPress 2, if you want to send the signal to the process group\n");

scanf("%d", &id);

if (id == 1) {

kill(pid, SIGINT);

} else if (id == 2) {

kill(-pid, SIGINT);

}

return 0;

}

Q3-A

202212012\_Lab8\_3\_receivesignal\_diffpgrp.c

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

void signal\_handler() {

printf("process %d got a SIGINT\n", getpid());

}

int main() {

signal(SIGINT, signal\_handler);

if (fork() == 0) {

setpgid(getpid(), 0);

printf("child pid %d and child group %d waiting\n", getpid(), getpgid(0));

} else {

}

pause();

return 0;

}

Output :



3.B

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

void signal\_handler() {

printf("process %d, %d got a SIGINT\n", getpid(), getpgid(0));

}

int main() {

signal(SIGINT, signal\_handler);

if (fork() == 0) { printf("child pid %d and child group %d waiting\n", getpid(), getpgid(0)); }

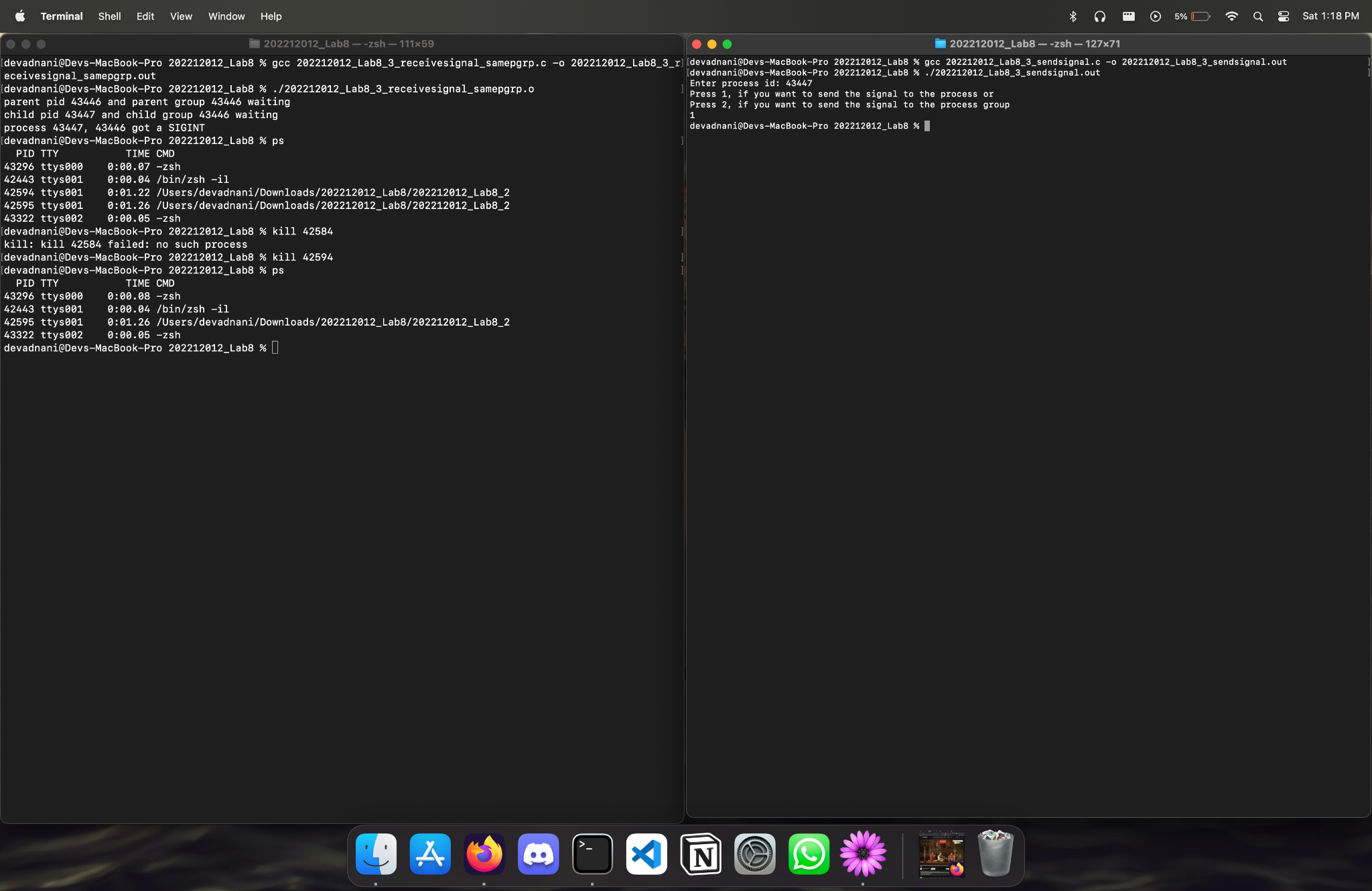
else { printf("parent pid %d and parent group %d waiting\n", getpid(), getpgid(0)); }

pause();

return 0;

}

Screenshot :



3:C

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

void signal\_handler() {

printf("process %d got a SIGINT\n", getpid());

}

int main() {

signal(SIGINT, signal\_handler);

if (fork() == 0) {

setpgid(getpid(), 0);

printf("child pid %d and child group %d waiting\n", getpid(), getpgid(0));

} else {

}

pause();

return 0;

}

Screenshot :

