Advanced Unix Programming
Assignment-9
Vijesh Ghandare – 111403013

Q1. Catch the SIGTERM signal, ignore SIGINT and accept the default action for SIGSEGV. Later let the program be suspended until it is interrupted by a signal. Implement using signal and sigaction.

CODE:

a9q1.c (~/Desktop/Lab9) - gedit

Open

```c
#include<stdio.h>
#include<signal.h>
#include<signal.h>
#include<stdlib.h>
#include<string.h>
static void sig_usr(int signo){
    if(signo == SIGTERM)
        printf("SIGTERM signal catched\n");
}
void quitproc(){
         printf("ctrl-\\ pressed to quit\n");
         exit(0); /* normal exit status */
}

int main(){
    if(signal(SIGINT,SIG_IGN) == SIG_ERR)
        printf("can't Ignore SIGINT");
    if(signal(SIGTERM,sig_usr) == SIG_ERR)
        printf("can't catch SIGTERM");
    if(signal(SIGSEGV,SIG_DFL) == SIG_ERR)
        printf("can't catch SIGSEGV");
    if(signal(SIGQUIT,quitproc) == SIG_ERR)
        printf("can't catch SIGQUIT");
    for(; ;)
        pause();
}
```

Output:

*** For signal SIGTERM & SIGINT**

```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ ./a.out &
[1] 3284
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ kill -15 3284
SIGTERM signal catched
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ fg % 1
./a.out
^C^C^C^C^C^C^C^C^C^C^C^C^C
^\ctrl-\ pressed to quit
```

***For signal SIGSEGV**

```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ ./a.out &
[1] 3292
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ ps
  PID TTY          TIME CMD
 3117 pts/2    00:00:00 bash
 3292 pts/2    00:00:00 a.out
 3293 pts/2    00:00:00 ps
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ kill -11 3284
bash: kill: (3284) - No such process
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ kill -11 3292
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ ps
  PID TTY          TIME CMD
 3117 pts/2    00:00:00 bash
 3305 pts/2    00:00:00 ps
[1]+  Segmentation fault      (core dumped) ./a.out
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$
```

Q2. Create a child process. Let the parent sleeps of 5 seconds and exits. Can the child send SIGINT to its parent if exists and kill it? Verify with a sample program.

CODE:

a9q2.c

```c
#include<stdio.h>
#include<signal.h>
#include<unistd.h>
#include<stdlib.h>
void sig_usr(int signo){
    if(signo == SIGINT)
    printf("\nSignal caught!");
    return;
}
int main(void){
    pid_t pid, ppid;
    if((pid = fork()) == 0){
        ppid = getpid();
        printf("ppid = %d\n", ppid);
        printf("Press ^c to kill parent..\n");
        kill(ppid, SIGINT);
        printf("After killing parent...\n");
    }
    else{
        printf("ppid-> %d pid-> %d ",ppid, pid);
        if(signal(SIGINT,sig_usr) == SIG_ERR)
            printf("Signal processed ");
        int time = sleep(5);
        printf("\nParent exiting with %d seconds left\n", time);
    }
    return 0;
}
```

OUTPUT:

```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ cc a9q2.c
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ ./a.out
ppid = 3622
Press ^c to kill parent..
^Cppid-> 0 pid-> 3622
Signal caught!
Parent exiting with 1 seconds left
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$
```

Q3. mplement sleep using signal function which takes care of the following:

a. If the caller has already an alarm set, that alarm is not erased by the call to alarm inside sleep implementation.

b. If sleep modifies the current disposition of SIGALRM, restore it

c. Avoid race condition between first call to alarm and pause inside sleep implementation using setjmp.

CODE:

a9q3.c (~/Desktop/Lab9) - gedit

Open

```c
#include<setjmp.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<signal.h>

static jmp_buf jb;

int sig_alrm(int signo){
    longjmp(jb, 1);
}

unsigned int sleep2(unsigned int secs){
    int time_left;
    time_left = alarm(0);       /* unslept seconds left from previous alarm */
    printf("unslept seconds from previous alarm %d\n", time_left);
    if(setjmp(jb) == 0){
        alarm(secs - time_left);    /* adding that time for user set value */
        pause();
    }
}

int main(){
    alarm(5);
    sleep2(7);
}

```

OUTPUT:

```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ cc a9q3.c
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ ./a.out
unslept seconds from previous alarm 5
Alarm clock
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$
```

Q4. “Child inherit parent’s signal mask when it is created, but pending signals for the parent process are not passed on”. Write appropriate program and test with suitable inputs to verify this.

CODE:

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
void err_sys(const char* x){
    perror(x);
    exit(1);
}
static void sig_quit(int signo){
    printf("caught SIGQUIT\n");
    if (signal(SIGQUIT, SIG_DFL) == SIG_ERR)
        err_sys("can't reset SIGQUIT");
}
void check_sigset(sigset_t sigset){
    int i;
    for(i = 0; i < 31; i++){
        if(sigismember(&sigset, i)){
            printf("SIGNAL %d present\n", i);
        }
    }
}
int main(void){
    sigset_t newmask, oldmask, pendmask, sigset;
    pid_t pid;

    if (signal(SIGQUIT, sig_quit) == SIG_ERR)
        err_sys("can't catch SIGQUIT");
    sigemptyset(&newmask);
    sigaddset(&newmask, SIGQUIT); // adding SIGQUIT to newmask

    if (sigprocmask(SIG_BLOCK, &newmask, &oldmask) < 0) // added SIGQUIT to BLOCK
        err_sys("SIG_BLOCK error");

    printf("Send SIGQUIT signals\n");
    sleep(5);
    /* SIGQUIT here will remain pending */
    if((pid = fork()) == -1){
        err_sys("Fork Error");
    }
    if(pid){
        printf("IN PARENT:\n");
```

```
41          if (sigprocmask(0, NULL, &sigset) < 0) {
42              err_sys("Error getting signal mask");
43          }
44          else {
45              check_sigset(sigset);
46          }
47
48          if (sigpending(&pendmask) < 0)
49              err_sys("signal pending error");
50
51          if (sigismember(&pendmask, SIGQUIT))
52              printf("IN PARENT: SIGQUIT pending\n");
53
54          wait();
55      }
56      else {
57          sigset_t childsigset;
58          printf("IN CHILD:\n");
59          if (sigprocmask(0, NULL, &childsigset) < 0) {
60              err_sys("Error getting signal mask");
61          }
62          else {
63              check_sigset(childsigset);
64          }
65
66          if (sigpending(&pendmask) < 0){
67              err_sys("sigpending error");
68          }
69
70          if (sigismember(&pendmask, SIGQUIT)){
71              printf("IN CHILD: SIGQUIT pending\n");
72          }
73          else {
74              printf("IN CHILD: SIGQUIT not pending in CHILD\n");
75          }
76      }
77  }
```

OUTPUT:

```
vijesh1996@vijesh1996-HP-Pavilion-15-Notebook-PC:~/Desktop/Lab9$ ./a.out
Send SIGQUIT signals
^\^\^\^\IN PARENT:
SIGNAL 0 present
SIGNAL 3 present
IN PARENT: SIGQUIT pending
IN CHILD:
SIGNAL 0 present
SIGNAL 3 present
IN CHILD: SIGQUIT not pending in CHILD
```

Explaination:
Above output shows that the quite signal pending in the parent is not pending in the child.