# CSE2005L - Operating Systems Lab

# Lab 4 - Interprocess Communication

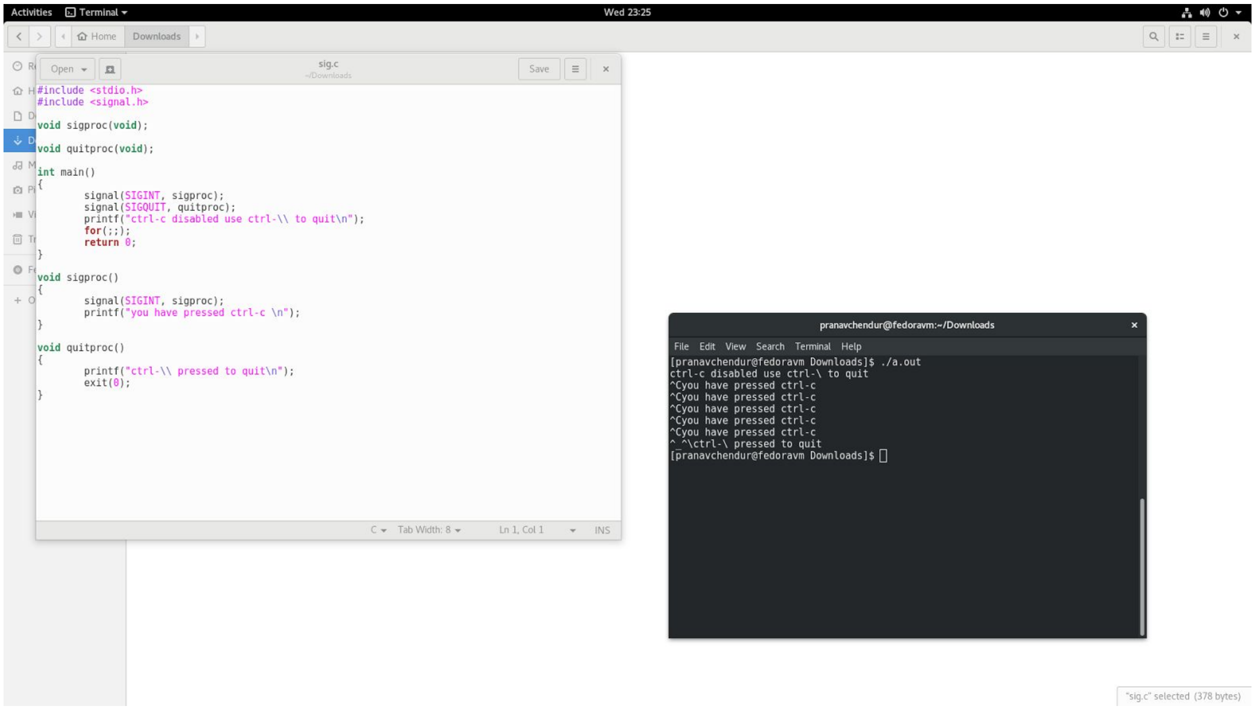
PRANAVCHENDUR T K - 15BCE1097

Faculty : Dr. Shyamala.L

## Code 1:

```
#include <stdio.h>
#include <signal.h>

void sigproc(void);

void quitproc(void);

int main()
{
      signal(SIGINT, sigproc);
      signal(SIGQUIT, quitproc);
      printf("ctrl-c disabled use ctrl-\\ to quit\n");
      for(;;);
      return 0;
}

void sigproc()
{
      signal(SIGINT, sigproc);
      printf("you have pressed ctrl-c \n");
}

void quitproc()
{
      printf("ctrl-\\ pressed to quit\n");
      exit(0);
}
```

**Output:**



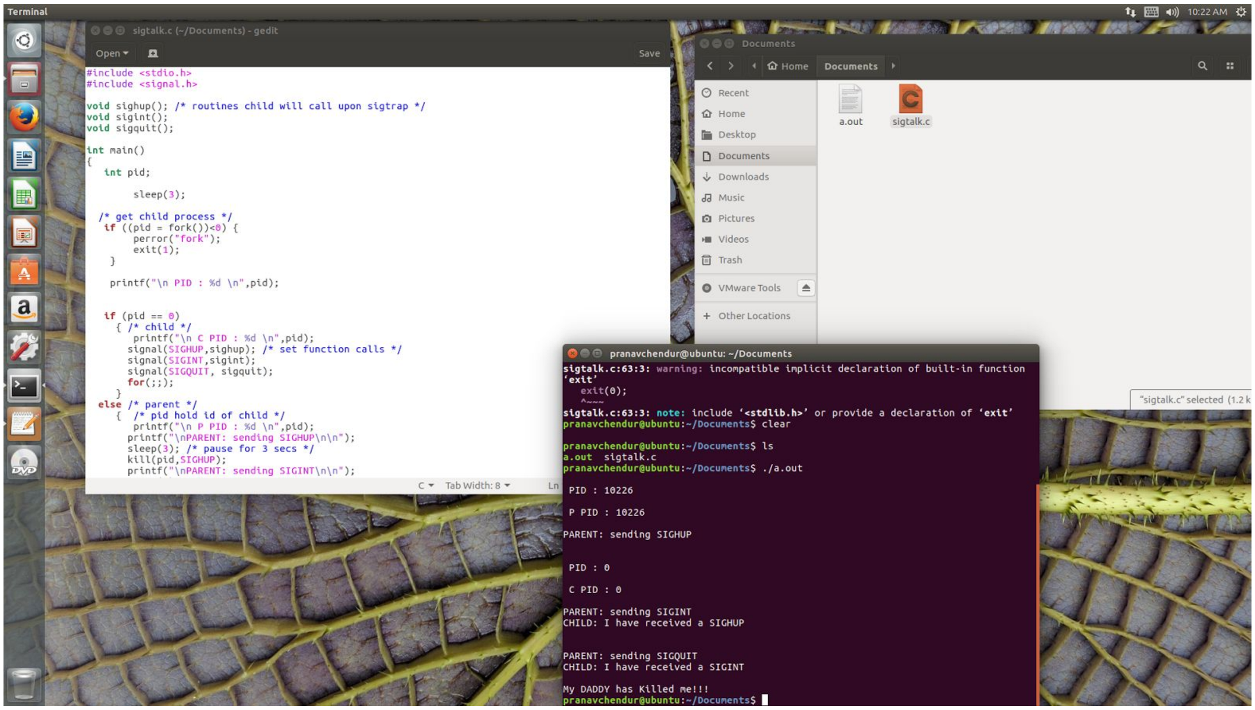
```c
#include <stdio.h>
#include <signal.h>

void sigproc(void);

void quitproc(void);

int main()
{
        signal(SIGINT, sigproc);
        signal(SIGQUIT, quitproc);
        printf("ctrl-c disabled use ctrl-\\ to quit\n");
        for(;;);
        return 0;
}

void sigproc()
{
        signal(SIGINT, sigproc);
        printf("you have pressed ctrl-c \n");
}

void quitproc()
{
        printf("ctrl-\\ pressed to quit\n");
        exit(0);
}
```

```
[pranavchendur@fedoravm Downloads]$ ./a.out
ctrl-c disabled use ctrl-\ to quit
^Cyou have pressed ctrl-c
^Cyou have pressed ctrl-c
^Cyou have pressed ctrl-c
^Cyou have pressed ctrl-c
^Cyou have pressed ctrl-c
^_^\ctrl-\ pressed to quit
[pranavchendur@fedoravm Downloads]$ 
```

## Code 2:

```c
#include <stdio.h>
#include <signal.h>

void sighup(); /* routines child will call upon sigtrap */
void sigint();
void sigquit();

int main()
{
   int pid;

      sleep(3);

  /* get child process */
   if ((pid = fork())<0) {
      perror("fork");
        exit(1);
    }

    printf("\n PID : %d \n",pid);
   if (pid == 0)
     { /* child */
      printf("\n C PID : %d \n",pid);
       signal(SIGHUP,sighup); /* set function calls */
       signal(SIGINT,sigint);
       signal(SIGQUIT, sigquit);
       for(;;);
     }
  else /* parent */
     {  /* pid hold id of child */
      printf("\n P PID : %d \n",pid);
       printf("\nPARENT: sending SIGHUP\n\n");
       sleep(3); /* pause for 3 secs */
       kill(pid,SIGHUP);
       printf("\nPARENT: sending SIGINT\n\n");
       sleep(3); /* pause for 3 secs */
       kill(pid,SIGINT);
       printf("\nPARENT: sending SIGQUIT\n\n");
       sleep(3);
       kill(pid,SIGQUIT);
     }

      return 0;
}

void sighup()

{  signal(SIGHUP,sighup); /* reset signal */
   printf("CHILD: I have received a SIGHUP\n");
}

void sigint()

{  signal(SIGINT,sigint); /* reset signal */
   printf("CHILD: I have received a SIGINT\n");
}

void sigquit()

{ printf("My DADDY has Killed me!!!\n");
  exit(0);
}
```

## Output :



```c
#include <stdio.h>
#include <signal.h>

void sighup(); /* routines child will call upon sigtrap */
void sigint();
void sigquit();

int main()
{
   int pid;

        sleep(3);

  /* get child process */
   if ((pid = fork())<0) {
        perror("fork");
        exit(1);
    }

    printf("\n PID : %d \n",pid);


   if (pid == 0)
     { /* child */
        printf("\n C PID : %d \n",pid);
       signal(SIGHUP,sighup); /* set function calls */
       signal(SIGINT,sigint);
       signal(SIGQUIT, sigquit);
       for(;;);
     }
  else /* parent */
     {  /* pid hold id of child */
        printf("\n P PID : %d \n",pid);
       printf("\nPARENT: sending SIGHUP\n\n");
       sleep(3); /* pause for 3 secs */
       kill(pid,SIGHUP);
       printf("\nPARENT: sending SIGINT\n\n");
```

```
sigtalk.c:63:3: warning: incompatible implicit declaration of built-in function
'exit'
   exit(0);
   ^~~~
sigtalk.c:63:3: note: include '<stdlib.h>' or provide a declaration of 'exit'
pranavchendur@ubuntu:~/Documents$ clear

pranavchendur@ubuntu:~/Documents$ ls
a.out  sigtalk.c
pranavchendur@ubuntu:~/Documents$ ./a.out

 PID : 10226

 P PID : 10226

PARENT: sending SIGHUP


 PID : 0

 C PID : 0

PARENT: sending SIGINT
CHILD: I have received a SIGHUP


PARENT: sending SIGQUIT
CHILD: I have received a SIGINT

My DADDY has Killed me!!!
pranavchendur@ubuntu:~/Documents$ 
```