PROGRAM 1

Activities Terminal Jul 25 14:02

Open *signalexample.c

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

void sig_handler(int signo)
{
    if(signo == SIGINT)
    printf("received SIGINT\n");
}

int main(void)
{
    if(signal(SIGINT , sig_handler) == SIG_ERR)
    printf("\n can't  catch  SIGINT\n");

    while(1)
        sleep(1);
      return 0;
}
```

C Tab Width: 8 Ln 19, Col 2 INS

Terminal

```
vebo-mcs@vebo-ubuntu-mcs:~$ gcc -o signalexample signalexample.c
vebo-mcs@vebo-ubuntu-mcs:~$ ./signalexample
^Creceived SIGINT
^Creceived SIGINT
^Creceived SIGINT
^Creceived SIGINT
^Creceived SIGINT
^Creceived SIGINT
```

PROGRAM EXPLANATION 1

sig_handler() This function is registered to the kernel by passing it as the second argument of the system call ‘signal’ in the main() function.

sleep(1) This function has been used in the while loop so that while loop executes after some time.

PROGRAM 2

Activities Terminal Jul 25 14:44

Open jumpsyscall.c Save

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

static jmp_buf jmpbuffer;

int main(int argc, char *argv[])
{
     int a;
     register b;
     volatile c;
        if (argc < 4)
        {
         printf("usage: %s a b c\n", argv[0]);
         exit(1);
        }
        a = atoi(argv[1]);
        b = atoi(argv[2]);
        c = atoi(argv[3]);

        printf("Before longjmp: a = %d, b = %d, c = %d\n", a, b, c);
        if (setjmp(jmpbuffer) != 0)
        {
           printf("After longjmp: a = %d, b = %d, c = %d\n", a, b, c);
           return 0;
        }

        a++;
        b++;
        c++;
        longjmp(jmpbuffer, 1);
        return 0;
}
```

Terminal

```
vebo-mcs@vebo-ubuntu-mcs:~$ ./jumpsyscall 1 2 3
Before longjmp: a = 1, b = 2, c = 3
After longjmp: a = 2, b = 3, c = 4
vebo-mcs@vebo-ubuntu-mcs:~$
```

C Tab Width: 8 Ln 23, Col 10 INS

PROGRAM EXPLANATION 2

Using atoi() we convert these argument into integers and set these values to variables a,b,c.after that we print the value of a,b and c we get a=1,b=2 and c=3.

initially setjmp() false hence control comes out from if condition and increase the value of variables a,b and c. longjmp() transfers control to the point set by setjmp()and it prints the value of a,b and c. hence a=2,b=3 and c=4 prints.

PROGRAM 3

Activities Text Editor Jul 25 15:25

Open chgfilepermission.c Save

```c
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
#include<time.h>

void printFileProperties(struct stat abc);

void main()
{
  char path [100];
  struct stat stats;
  printf("Enter the source file");
  scanf("%s",path);

     //
     if (stat(path,&stats )==0)
     {
         printFileProperties(stats);
     }
     else
     {
        printf("Unable to get File Properties.\n");
        printf("Please check whether '%s' file exists.\n",path);
     }
}


void  printFileProperties (struct stat stats)
{
       struct tm dt;
        printf("\nFileAccess");

        if(stats.st_mode & R_OK)
```

C Tab Width: 8 Ln 1, Col 1 INS

Activities Terminal Jul 25 15:26

Open chgfilepermission.c Save

```c
    }
}


void  printFileProperties (struct stat stats)
{
       struct tm dt;
        printf("\nFileAccess");

        if(stats.st_mode & R_OK)
                printf("\tread");

        if(stats.st_mode & R_OK)
                printf("\twrite");


        if(stats.st_mode & R_OK)
                printf("\texecute");

        //file size
        printf("\n File Size: %ld",stats.st_size);


        // Get File  creation  time  in seconds  and
        // Convert  seconds   to date  and time  format

        dt=*(gmtime(&stats.st_ctime));
        printf("\nCreated on: %d-%d-%d  %d:%d:%d",
         dt.tm_mday , dt.tm_mon , dt.tm_year +1900, dt.tm_hour , dt.tm_min , dt.tm_sec);



        // File  modification time
        dt =*(gmtime(&stats.st_mtime));
        printf("\nModified on : %d-%d-%d  %d:%d:%d" ,
        dt.tm_mday, dt.tm_mon , dt.tm_year + 1900  ,dt.tm_hour , dt.tm_min , dt.tm_sec);

}
```

Terminal

```
vebo-mcs@vebo-ubuntu-mcs:~$ gcc -o chgfilepermission chgfilepermission.c
vebo-mcs@vebo-ubuntu-mcs:~$ ./chgfilepermission
Enter the source file     name.txt

FileAccess      read    write   execute
 File Size: 25
Created on: 25-6-2021  9:51:39
vebo-mcs@vebo-ubuntu-mcs:~$
```

C Tab Width: 8 Ln 37, Col 35 INS

PROGRAM EXPLANATION 4

stat() function is used to list properties of a file identified by path
void printFileProperties(struct stat stats) function is used to print file properties

i.e File access,File size,file creation date and time,File modification date time

PROGRAM 5

Activities Text Editor Jul 25 17:47

Open test.c Save

```c
#include<string.h>
#include<stdio.h>
#include<stdlib.h>


struct student
{
        int  stud_id;
        int name_len;
        int struct_size;
        char stud_name[0];   //variable length array must be last.
};

//Memory allocation and initialisation of structure
struct student *createStudent(struct student *s, int id, char a[])
{
        s=malloc(sizeof(*s) + sizeof(char) *strlen(a));

        s->stud_id =id;
        s->name_len=strlen(a);
        strcpy(s->stud_name,a);

        s->struct_size =(sizeof(*s) +sizeof(char) * strlen(s->stud_name));

        return s;
}

//  Print student  details


void  printStudent(struct student *s)
{
   printf("Student_id    : %d\n"
          "Stud_Name     : %s\n"
          "Name_Length   : %d\n"
          "Allocated_Struct_size: %d\n\n",
          s->stud_id ,s->stud_name , s->name_len , s->struct_size);
```

C Tab Width: 8 Ln 6, Col 2 INS

Activities Terminal Jul 25 17:48

Open test.c Save

```c
//  Print student  details



void  printStudent(struct student *s)
{
   printf("Student_id     : %d\n"
          "Stud_Name      : %s\n"
          "Name_Length    : %d\n"
          "Allocated_Struct_size: %d\n\n",
          s->stud_id ,s->stud_name , s->name_len , s->struct_size);

          //Value of Allocated_Struct_size here  is in bytes.
}

//Driver code

int main()
{
        struct student  *s1 , *s2;
        s1=createStudent(s1 , 523,"Bablu");
        s2=createStudent(s2 , 535,"Guddu");

        printStudent(s1);
        printStudent(s2);

        //size in bytes
        printf("Size of struct student %lu\n", sizeof(struct student));
        //size in bytes
        printf("Size of struct student %lu\n", sizeof(s1));

        return 0;
}
```

Terminal

```
vebo-mcs@vebo-ubuntu-mcs:~$ gcc -o test test.c
vebo-mcs@vebo-ubuntu-mcs:~$ ./test
Student_id      : 523
Stud_Name       : Bablu
Name_Length     : 5
Allocated_Struct_size: 17

Student_id      : 535
Stud_Name       : Guddu
Name_Length     : 5
Allocated_Struct_size: 17

Size of struct student 12
Size of struct student 8
vebo-mcs@vebo-ubuntu-mcs:~$
```

C Tab Width: 8 Ln 6, Col 2 INS

PROGRAM EXPLANATION 5

Structure – Structure is a user defined datatype which can hold different types of elements. In the above program we create a studentstructure which holds student data. We create two variables of structre s1 and s2. Inside main mehod we create a method createstudent() and pass the student values. Now control comes in createstudent(). Inside this function we allocate the memory for each structure variable using malloc(). Createstudent() returns student details and using printstudent() we display the student details

PROGRAM 6

Activities Terminal Jul 25 18:08

Open pipexample.c Save

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


int main()
{
        int pipefds[2];
        int returnstatus ;
        char writemessages[2][20]={"Hi","Hello C"};
        char readmessage[20];
        returnstatus = pipe(pipefds);

        if(returnstatus ==-1)
        {
         printf("Unable to create pipe\n");
         return 1;
        }

        printf("Writing to pipe   -Message 1 is %s\n",writemessages[0]);
        write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
        read(pipefds[0], readmessage, sizeof(readmessage));
        printf("Reading  to pipe   -Message 1 is %s\n",readmessage);
        printf("Writing to pipe   -Message 2 is %s\n",writemessages[0]);

        write(pipefds[1], writemessages[1], sizeof(writemessages[0]));
        read(pipefds[0], readmessage, sizeof(readmessage));
        printf("Reading  to pipe   -Message 2 is %s\n",readmessage);

        return 0;
}
```

Terminal

```
vebo-mcs@vebo-ubuntu-mcs:~$ gcc -o pipeexample pipexample.c
vebo-mcs@vebo-ubuntu-mcs:~$ ./pipeexample
Writing to pipe   -Message 1 is Hi
Reading  to pipe   -Message 1 is Hi
Writing to pipe   -Message 2 is Hi
Reading  to pipe   -Message 2 is Hello C
vebo-mcs@vebo-ubuntu-mcs:~$
```

C Tab Width: 8 Ln 1, Col 1 INS

PROGRAM EXPLANATION 6

pipe(pipefds): This system call would create a pipe for one-way communication i.e., it creates two descriptors, first one is connected to read from the pipe and other one is connected to write into the pipe.

PROGRAM 7

setjumpexample.c

```c
#include<stdio.h>
#include<setjmp.h>
jmp_buf buf;

int main()
{
  int x=1;
  int i=1;
  setjmp(buf);

   printf("%d :5\n",i);
  i++;
  x++;

  if(x<=100)
      longjmp(buf,1);

 return 0;
}
```

Terminal

```
vebo-mcs@vebo-ubuntu-mcs:~$ gcc -o setjumpexample setjumpexample.c
vebo-mcs@vebo-ubuntu-mcs:~$ ./setjumpexample
1 :5
2 :5
3 :5
4 :5
5 :5
6 :5
7 :5
8 :5
9 :5
10 :5
11 :5
12 :5
13 :5
14 :5
15 :5
16 :5
17 :5
18 :5
19 :5
20 :5
21 :5
22 :5
```

Activities Terminal Jul 25 18:18

setjumpexample.c

```c
#include<stdio.h>
#include<setjmp.h>
jmp_buf buf;

int main()
{
  int x=1;
  int i=1;
  setjmp(buf);

   printf("%d :5\n",i);
  i++;
  x++;

  if(x<=100)
      longjmp(buf,1);

 return 0;
}
```

Terminal

```
78 :5
79 :5
80 :5
81 :5
82 :5
83 :5
84 :5
85 :5
86 :5
87 :5
88 :5
89 :5
90 :5
91 :5
92 :5
93 :5
94 :5
95 :5
96 :5
97 :5
98 :5
99 :5
100 :5
vebo-mcs@vebo-ubuntu-mcs:~$
```

C Tab Width: 8 Ln 8, Col 10 INS

PROGRAM EXPLANATION 7

setjmp(buf) This system call is used to set the jump position using buf.

longjmp() This system call is used to Jump to the point located by setjmp.

The longjump() transfers control the pointe which is pointed by setjump().this process will be terminate after the terminating loop.

PROGRAM 8

Activities Terminal Jul 25 18:28

forkex1.c

```c
#include <stdio.h>
#include <sys/types.h>
int fork();
void main(int argc, char* argv[])
{
        fork();
        fork();
        fork();

        printf("hello\n");
}
```

Terminal

```
vebo-mcs@vebo-ubuntu-mcs:~$ gcc -o forkex1 forkex1.c
vebo-mcs@vebo-ubuntu-mcs:~$ ./forkex1
hello
hello
hello
hello
hello
vebo-mcs@vebo-ubuntu-mcs:~$ hello
hello
hello
```

C Tab Width: 8 Ln 6, Col 5 INS

PROGRAM EXPLANATION 8

The number of times ‘hello’ is printed is equal to number of process created. Total Number of Processes = 2^fork calls.

So here fork calls = 1, 2^1 = 2. Hence Hello World print 2 times.

PROGRAM EXPLANATION 9

Activities Terminal Jul 25 18:46

Open *statuschildps.c Save

```c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>

//Driver Code
int main(void)
{
   pid_t pid=fork();
   if(pid == 0)
   {
       execl("/bin/sh" , "bin/sh", "-c" , "./nopath" , (char *)0);
       // don't use NULL it throws warning of missing sentinel in function call because
       /**
It looks like you may not have terminated an array declaration with NULL. Without the null you may have some memory weirdness as the runtime won't know where the array ends and the next bit of memory starts.
Instead use (char *)0     **/
   }

    int status;
    waitpid(pid, &status ,0);

    if(  WIFEXITED(status))
    {
      int exit_status = WEXITSTATUS(status);
      printf("Exit status of the child was %d\n",exit_status);
    }
    return 0;
}
```

Terminal

```
schildps.c
vebo-mcs@vebo-ubuntu-mcs:~$ ./statuschildps
bin/sh: 1: ./nopath: not found
Exit status of the child was 127
vebo-mcs@vebo-ubuntu-mcs:~$
```

PROGRAM EXPLANATION 9

execl() system function takes the path of the executable binary file
fork() system call is used to create a new process which becomes child of the caller process.

WIFEXITED(status) : returns true if the child terminated normally.
waitpid() system call : It suspends execution of the calling process until a child specified by pid argument has changed state.

PROGRAM 10

Activities Terminal Jul 25 18:59

Open cpyfile2file.c Save

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

int main()
{
   FILE *fptr1, *fptr2;
   char filename[100], c;
   printf("Enter the filename to open for reading \n");
   scanf("%s", filename);
   fptr1 = fopen(filename, "r");

   if(fptr1 == NULL)
   {
      printf("Cannot open file %s \n", filename);
      exit(0);
   }

   printf("Enter the filename to open for writing \n");
   scanf("%s", filename);

   // Open another file for writing
   fptr2 = fopen(filename, "w");
   if(fptr2 == NULL)
   {
      printf("Cannot open file %s \n", filename);
      exit(0);
   }


   //Read the contents of the file

   c= fgetc(fptr1);
   while(c!=EOF)
   {
     fputc(c,fptr2);
     c =fgetc(fptr1);
   }

   printf("\n Contents copied to %s", filename);
   fclose(fptr1);
   fclose(fptr2);
   return 0;
}
```

Terminal

```
vebo-mcs@vebo-ubuntu-mcs:~$ gcc -o cpyfile2file cpyfile2file.c
vebo-mcs@vebo-ubuntu-mcs:~$ ./cpyfile2file
Enter the filename to open for reading
name.txt
Enter the filename to open for writing
cpy.txt

 Contents copied to cpy.txtvebo-mcs@vebo-ubuntu-mcs:~$ cat cpy.txt
HELLO it's Vaibhav Joshi
vebo-mcs@vebo-ubuntu-mcs:~$
```

C Tab Width: 8 Ln 3, Col 1 INS

PROGRAM EXPLANATION 10

fopen(filename, "r") this function is used to open file for reading.
fopen(filename, "w") this function is used to open file for writing.
fgetc(fptr1) read contents from file.
fputc(c, fptr2) write contents to the file.
fclose(fptr1) this function is used for close a file.