WEEK 10

1.Write a C program that illustrates suspending and resuming processes using signals

Program

```
#include <stdio.h>
#include <ospace/unix.h>
int child_function()
{
  while (true) // Loop forever.
    {
    Printf("Child loop\n");
    os_this_process::sleep( 1 );
    }
  return 0; // Will never execute.
}
int main()
{
  os_unix_toolkit initialize;
  os_process child ( child function ); // Spawn child.
  os_this_process::sleep( 4 );
  printf("child.suspend()\n");
  child.suspend();
  printf("Parent sleeps for 4 seconds\n");
  os_this_process::sleep (4);
  printf("child.resume()");
  child.resume ();
  os_this_process::sleep (4);
  printf("child.terminate()");
  child.terminate ();
  printf("Parent finished");
  return 0;
}
```

Output

```
Child loop
Child loop
Child loop
Child loop
Child loop
child.suspend()
Parent sleeps for 4 seconds
child.resume()
Child loop
Child loop
Child loop
Child loop
child.terminate()
Child loop
Parent finished
```

2.Write client and server programs(using c) for interaction between server and client processes using Unix Domain sockets.

Program

```
#include <stdio.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <string.h>
int main(void)
{
 struct sockaddr_un address;
 int  socket_fd, nbytes;
 char buffer[256];
 socket_fd = socket(PF_UNIX, SOCK_STREAM, 0);
 if(socket_fd < 0)
 {
  printf("socket() failed\n");
  return 1;
 }
 /* start with a clean address structure */
 memset(&address, 0, sizeof(struct sockaddr_un));
 address.sun_family = AF_UNIX;
 snprintf(address.sun_path, UNIX_PATH_MAX, "./demo_socket");
 if(connect(socket_fd,
 (struct sockaddr *) &address,
 sizeof(struct sockaddr_un)) != 0)
 {
  printf("connect() failed\n");
  return 1;
 }
 nbytes = snprintf(buffer, 256, "hello from a client");
 write(socket_fd, buffer, nbytes);
 nbytes = read(socket_fd, buffer, 256);
 buffer[nbytes] = 0;
 printf("MESSAGE FROM SERVER: %s\n", buffer);
 close(socket_fd);
 return 0;
```

Output

```
Student@ubuntu:~$gcc –o server1.out server1.c
Student@ubuntu:~$gcc client1.out client1.c
Student@ubuntu:~$ ./client1.out
MESSAGE FROM SERVER hello from the server
Student@ubuntu:~$ ./server1.out
MESSAGE FROM CLIENT hello from the CLIENT
```

THANK YOU