LAB 01: MULTIPROCESSING

GUNARATNE D.L,

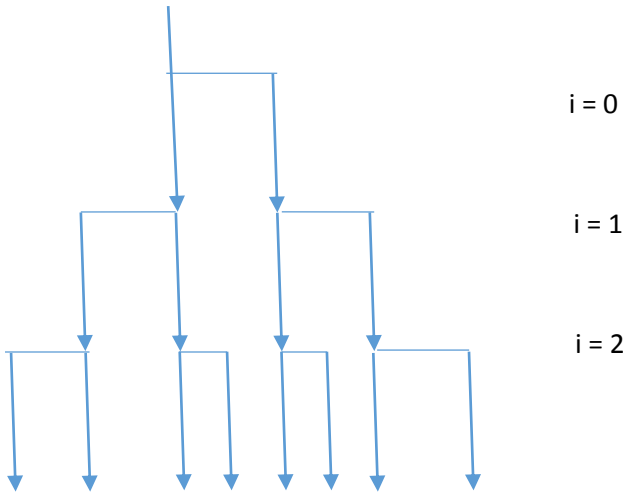
E/11/133,

23.07.2015

01) ii) init

02) i) Parent then Child. Yes, because now there will be 2 processes containing same code.

ii) Parent & 7 children



03)ii)

```
int main(void)
{
int pid;
pid = fork();


        if (pid < 0)
        {
                perror("fork");
                exit(1);
        }else if(pid > 0 ){

                wait();

        }
        if (pid == 0)
                puts("This is the child process");
        else
                puts("This is the parent process");
```

```
        return 0;
}
```

04)i) That will not be printed.

ii)

```
//Simple shell
#include<stdlib.h>
#include<stdio.h>
#include<unistd.h>

int main(){
      int pid;
      char arg1[200];
      char arg2[200];
      while(1){

            pid = fork();
            wait();

            if(pid < 0){
                  puts("fork error");
            }

            if(pid == 0){

                  puts("\nEnter a command:") ;
                  gets(arg1);
                  puts("\nEnter the path:") ;
                  gets(arg2);

                  execl(arg1, "-l",arg2, NULL);
            }
            //execl(arg);

      }
}
```

05)

1)

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(){

int sockfd,newsockfd,n,pid,listenfd;
```

```
struct sockaddr_in servaddr,cli_addr;
socklen_t clilen;
char* banner = "Hello TCP client! This is TCP server";
char buffer[100];

/* one socket is dedicated to listening */
sockfd=socket(AF_INET,SOCK_STREAM,0);

/* initialize a sockaddr_in struct with our own address information for
binding the socket */
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(12345);

/* binding */
bind(sockfd,(struct sockaddr *)&servaddr,sizeof(servaddr));
listen(sockfd,5);
clilen = sizeof(cli_addr);

while (1)
{
/* New socket descriptor is returned each time a client connects*/
        newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
if (newsockfd < 0)
{
        perror("ERROR on accept");
        exit(1);
}
pid = fork();
if (pid < 0)
{
        perror("ERROR on fork");
        exit(1);
}
if (pid == 0)
{
/* In child process which the handles client connection */
        close(sockfd);
        n = recvfrom(newsockfd,buffer,1000,0,(struct sockaddr *)&cli_addr,&clilen);//information of
the client by recvfrom function
        buffer[n] = 0;
        sendto(newsockfd,banner,strlen(banner),0,(struct sockaddr*)&cli_addr,sizeof(cli_addr));
        printf("Received:%s\n",buffer);
        exit(0);
```

```
}
else
/* In parent process which continues to listen for new clients */
        close(newsockfd);
}
}
```

2) Server will wait till first connection is closed in other word it will wait till 1st child's work is finished.

3) Clients can't connect to the server.

4)Yes, They can.

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>




int main(){

int sockfd,newsockfd,n,pid;
struct sockaddr_in servaddr,cli_addr;
socklen_t clilen;
char* banner = "Hello TCP client! This is TCP server.Send the
file location :\n";
char buffer[100],fileContent[20000];


/* one socket is dedicated to listening */
sockfd=socket(AF_INET,SOCK_STREAM,0);

/* initialize a sockaddr_in struct with our own address
information for
binding the socket */
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(12345);

/* binding */
bind(sockfd,(struct sockaddr *)&servaddr,sizeof(servaddr));
listen(sockfd,5);
clilen = sizeof(cli_addr);
```

```
while (1)
{
/* New socket descriptor is returned each time a client
connects*/
      newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr,
&clilen);

      if (newsockfd < 0)
      {
            perror("ERROR on accept");
            exit(1);
      }
      pid = fork();
      if (pid < 0)
      {
            perror("ERROR on fork");
            exit(1);
      }
      if (pid == 0)
      {
      /* In child process which the handles client connection */

            close(sockfd);

            sendto(newsockfd,banner,strlen(banner),0,(struct
sockaddr*)&cli_addr,sizeof(cli_addr));
            n = recvfrom(newsockfd,buffer,1000,0,(struct sockaddr
*)&cli_addr,&clilen);//information of the client by recvfrom
function
            buffer[n-1] = 0;

      //    Reading the file

            FILE *fp;

               fp = fopen(buffer,"r"); // read mode
               printf("The contents of %s file are :\n", buffer);

               if( fp == NULL )
               {
                  perror("Error while opening the file.\n");
                  exit(EXIT_FAILURE);
               }



               int size = 0;

               char ch;

               while( ( ch = fgetc(fp) ) != EOF ){
                       printf("%c",ch);
```

```
                        fileContent[size] = ch;
                    size++;

                }
                fclose(fp);


            sendto(newsockfd,fileContent,size+1,0,(struct
sockaddr*)&cli_addr,sizeof(cli_addr));
            printf("Received:%s\n",buffer);
            exit(0);
      }
      else
      /* In parent process which continues to listen for new
clients */
            close(newsockfd);

}
}
```

All relevant programs are available in this folder.

Problem 04 : shellTest2.c

Problem 05 : serever0.c & server.c