Name:……

Student ID:……

Birzeit University - Faculty of Information Technology

Computer Systems Engineering Department – ENCS531
2nd semester - 2012/13 - first hour exam – duration (80 minutes)- 19 MAR 2013

Real-Time Applications and Embedded Systems

Instructor: Dr. Ahmad Afaneh

Q1. Select the most correct answer. Write down your final answers in the following table. (30 points)

1	2	3
		1. 2.
4	**5**	**6**

1) **The difference between the process and the program is**

a. They are the same
b. The process includes the program
c. The program includes the process
d. none of the above

2) **When a signal is masked (blocked) it will**

a. never be delivered
b. be delivered once it is unblocked
c. still cause an interrupt
d. none of the above

3) **The essential IPC communication models are**

1. Shared memory
2. Messages

4) **Which statement is false regarding Named Pipes**

a. exist as device special files
b. managed by the OS
c. they only work for parent child communication

5) **Mailslot is used for two way inter-process communications.**

a. True
b. False

6) **What is the main difference in semaphore implementation between Linux and Windows ? linux defines semaphores as sets**

Name:……

Student ID:……

Q2. (70 points) In preparations for the next code competition we decided to write our own automated judging system (a system that will decide if the code is correct or not). The system contains a client, server and other processes. Your task is to just write the server. The server job is to receive the source code, make sure it is safe to run, compile it and test it using predefined tests. The following are the main steps

1. *The server is initialized and ready to accept connections*
2. *The client connects to the server* ***[15 points]***
3. *The client sends the source code to the server* ***[10 points]***
4. *The server sends the file name to another process on the server (policyCheck) using any IPC*
5. *policyCheck then sends either 0 or 1 as a response to the server (0: code is not safe the server sends an error to the client, 1: the code is safe the server continues* ***[10 points]***
6. *the server compiles the code (hint: it calls gcc)* ***[5 points]***
7. *the server has test files input.txt and output.txt, the server runs the code with input.txt as input and compares the output with output.txt (hint use < and > to redirect input and output)* ***[5 points]***
8. *the server has a function checkOutput (char* file1,char* file2) that returns true if the two files match 0 otherwise. (Don't write this function assume it exists already)* ***[5 points]***
9. *the server sends the results report to the client*
10. *connection closed* ***[5 points]***

your task is to write the sever process , please keep in mind

- ***the server should support multiple clients [10 points]***
- ***the details of messages and communication between the processes is not strict***
- ***the result report format is not strict***

code structure [5 points]

////////////// ***step 1,2*** ////////////////////////////////

Name:……

Student ID:……

Int main ()

```
{
   int srvfSoc;
  struct sockaddr_in srv; /* used by bind() */
  /* create socket */
 if((srvfSoc = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
      perror("socket");
     exit(1);
   }
srv.sin_family = AF_INET; /* use the Internet addr family */
srv.sin_port = htons(2345); /* bind socket 'fd' to port 2345*/
/* bind: a client may connect to any of my addresses */
srv.sin_addr.s_addr = htonl(INADDR_ANY);
if(bind(srvfSoc, (struct sockaddr*) &srv, sizeof(srv)) < 0) {
     perror("bind"); exit(1);
    }
if(listen(srvfSoc, 5) < 0) {
   perror("listen");
   exit(1);
  }
While(1)
{/* accept new client*/
struct sockaddr_in cli; /* used by accept() */
int* newfd; /* returned by accept() */
int cli_len = sizeof(cli);
pthread_t f_thread;
newfd= (int*)malloc(sizeof(int));
*newfd = accept(fd, (struct sockaddr*) &cli, &cli_len);
if(*newfd < 0) {
perror("accept"); exit(1);
}
/* create new thread to handle each client*/
If (!fork())
 {
  processClient (newfd);
 return 0;
  }
}
```

} // end of main

```
Void  ProcessClient (int* fd)
{
int clientfd=*fd;
free (fd);
int ver;
int nbytes;

int reply;

//////////////  step 3 uploading the file //////////////////

Int filed;

filed= open(FILENAME, O_CREAT| O_WRONLY);

while (1)
{ /*read from server */
  if((nbytes = read(clientfd, buf, BUF_SIZE)) < 0) {
      perror("read"); exit(1);}
//write to file
write(filed, buf,nbytes);
if (nbytes <BUF_SIZE) // if end of file
  {
    close(filed);
    close(clientfd);
     return;
   }

}

// //////////////step 4,5 ///////////////////////

/// use any IPC to send filename and get reply////////////

write(clientFd, &reply,sizeof(int));

If(! reply)

  {  close (clientfd);

     return;

}

/// step 6/////////////////

If(!fork())

{

execlp(" /usr/bin ","gcc" , FILENAME);

}else wait(NULL);
```

```
/// step 7/////////////////

If(!fork())

{

execlp(" /usr/bin ","a.out" , "<","input.txt",">","out.txt");

}else wait(NULL);

/// step 8,9///////////////////////

If (CheckOutput(output.txt,out.txt))

{ // send client a pass report

 Write (clientfd,…);

} else

// send client a fail report

 Write (clientfd,…);

/// step 10////////////////////

Close(clientfd);e

}
```