2003- SEPTIEMBRE

AMPLIACION DE SISTEMAS OPERATIVOS 5/09/2003

Alumno/a D./Dña.....

Ejercicio 1.- (3 puntos) Analice el siguiente código respondiendo a las siguientes preguntas y justificando las respuestas: ¿Resuelve algun problema de sincronización entre procesos? ¿Qué metodología utiliza? Analice y explique las tareas ¿Qué misión tienen? ¿Existen colas implícitas en las tareas? ¿Dónde? ¿De qué tipo son? En el lenguaje ADA95 que misión tiene la construcción select. Ponga un ejemplo de su funcionalidad

```
task Server is
        entry Request (A: in Address; X: in Item);
end;

task body Server is
Reply: Address;
Job: Item;
  begin
   loop
    accept (A: in Address; X: in Item) do
    Reply: := A;
    Job:= X;
  end;
.....trabaja sobre Job
Reply.Deposit(Job);
  end loop;
end Server;
```

```
task User;
task body User is
                My_Box: Address := new Mailbox;
                My_Item: Item;
        begin
                Server.Request (My_Box; My_Item);
                ..... Hace otras cosas mientras espera
        My_Box.Collect ( My_Item);
end User;
```

Ejercicio 2.- (3 puntos). Analice el siguiente código y responda a las siguientes preguntas, justificando las respuestas: ¿Cuál es la misión de los sockets o hilos, si es que aparecen en el código? ¿Qué diferencias existen entre procesos e hilos? ¿Los sockets son clientes o servidores? ¿Qué tipo de constructor de socket utiliza? ¿Resuelve algún tipo de problema de sincronización, de comunicación o de sección crítica? ¿En su caso, como lo resuelve?

```
import java.net.*;
import java.io.*;
import java.util.*;

public class jhttp extends Thread {
        Socket theConnection;
        static File docroot;
        static String indexfile = "index.html";

public jhttp(Socket s) {
        theConnection = s;
}


public static void main (String[] args) {

        int thePort;
        ServerSocket ss;
        try {

        docroot = new File(args[0]);
        }
catch (Exception e) {
docroot = new File (".");
}
```

```
try {
        thePort = Integer.parseInt(args[1]);
        if (thePort < 0 || thePort > 65535) thePort =
80;
        }
        match (Exception e) {
                thePort = 80;
        }
        try {
        ss= new ServerSocket (thePort);
        System.out.println("Acepting connections
on port" + ss.getLocalPort());
        System.out.println("Document Root:" +
docroot);
        while (true) {
                jhttp j = new jhttp(ss.accept());
        j.start();
        }
}
catch (IOException e) {
System.err.println("Servidor caído
prematuramente");
}
}
public void run() {
.....
}
```

2003 – SEPTIEMBRE

Ejercicio 3.- (4 puntos)

a) Analice el siguiente código formado por dos programas, indicando cual es el cometido de cada uno. ¿Qué código tiene características de padre o de hijo? ¿Es un código que se puede distribuir? ¿Cuántos procesos se pueden distribuir? ¿Cómo hacerlo en dos máquinas concretas?

b) Analice con detalle las funciones de programación en grupo que existen. ¿De que tipo son?

c) Indique que misión tienen, en este código, todas las demás funciones de pvm (sin repetirlas) que aparecen en el mismo.

d) ¿Es parte de la resolución de un problema de Comunicación, Sincronización o Sección Crítica? Justifique la respuesta

printer.c

```
#include <stdio.h>
#include <stdlib.h>
#include "pvm3.h"
#define MSGTAG_ESPECIALES 1
#define MSGTAG_NORMALES 2
#define MSGTAG_PADRE 10
main(int argc, char *argv[])
{
        int n_esp, n_norm;
        int k;
        int *tids_esp, *tids_norm;
        int norm_expand, esp_expand;
        int inf_buf, inf;
        char nombre_host[20];
        pvm_catchout(stdout);
        n_esp=atoi(argv[1]);
        n_norm=atoi(argv[2]);
        k=atoi(argv[3]);
        if (n_esp<0 || n_norm<0 || k<0 || n_esp<k || k==0 && n_esp>0)
        {
                if(n_esp<k)
                        printf("\nEl tercer parámetro ha de ser menor que el primero\n");
                else
                {
                        if (k==0 && n_esp > 0)
                            printf("\nEl tercer parámetro no puede ser 0\n");
                        else
                            if(n_esp<0 || n_norm<0 || k<0)
                               printf("\nNingún valor de entrada puede ser negativo\n");
                }
                pvm_exit();
                exit(0);
        }
        if(n_esp!=0)
        {
                if ((tids_esp = (int *) (malloc(sizeof(int)*n_esp))) == NULL)
                {
                        pvm_perror("\nFallo en Reserva de Memoria de Especiales\n");
                        pvm_exit();
                        exit(0);
                }
                esp_expand=pvm_spawn("esp", (char**)0, 0, "", n_esp, tids_esp);

                if(esp_expand!=n_esp)
                {
                        pvm_perror("<< Padre >> Error en expansión de trabajos especiales");
                        pvm_exit();
                }
        }
        if(n_norm!=0)
        {
```