**Universidad Tecnológica Nacional**
**Facultad Regional Buenos Aires**
**Departamento de Ingeniería Electrónica**

**Informática I**
24 de Septiembre de 2021
**Final**

| Apellido y Nombres | Legajo | Profesor | # Hojas |
|---|---|---|---|
| | | | |

**Normas Generales**

Lea detenidamente cada pregunta y consulte las dudas de interpretación que pudiesen surgir.

## **Sección teórica** (tiempo estimado 30 minutos)

1. ¿Qué tamaño ocupa en memoria?

```
union info1{
      float vector[3];
      char str[10];
      int *p;
      };
```

En una máquina con enteros de 32bits y punteros de 64bits.

2. ¿Cuáles son las 3 formas en las que se puede manejar el cierre de un proceso hijo desde el padre de forma de evitar los procesos zombies?

3. La función fork(), ¿Para qué sirve? ¿Qué devuelve como valor de retorno?

4. ¿Cuál es la diferencia entre usar struct o union? ¿En que casos usaría union?

5. ¿En que situaciones el uso de un puntero a función nos puede ayudar? De un ejemplo practico de uso de punteros a función.

***Notas para la parte teórica:***
Se deberá entregar un archivo de texto donde indique el punto y la o las letras de las respuestas que considere correctas.
Al menos debe tener 6 puntos para poder aprobar. Cada pregunta vale 2 puntos si la respuesta es correcta y 1 punto si la respuesta no es totalmente satisfactoria.

| Apellido y Nombres | Legajo | Profesor | # Hojas |
|---|---|---|---|
| | | | |

## **Sección práctica** (tiempo estimado 1 hora y 30 minutos)

1. Realizar un programa que capture la información de carga del sistema (loadavg) mediante la información provista por el archivo `/proc/loadavg`:

```
0.57 0.49 0.39 1/1106 6472
```

De esta cadena de caracteres (separada por espacios) deberá tomar:

- El primer valor (promedio de un 1 minuto del uso del procesador): `0.57`
- Y al cuarto valor dividirlo en dos partes:
  - La cantidad de procesos concurrentes: `1`
  - La cantidad de procesos en el sistema: `1106`

El programa deberá:

a. Leer este archivo una vez por minuto hasta completar 15 lecturas. En cada lectura deberá abrir y cerrar el archivo (para que se actualice la información). Cada uno de estos elementos (la estructura con los 3 datos antes mencionados: float, int, int) se deberá guardar en un vector (lista compacta) y superada su capacidad deberá re-escribir los del principio.

b. Cada 5 minutos se deberá sacar el promedio de estos 15 datos. Con `mytime()` deberá conseguir la hora (devuelve una cadena de texto del tipo AAAAMMDDHHMM, por ejemplo 202109241245) y enviarla junto al promedio a una FIFO. Resultando una cadena separada con punto y coma (;) comenzando con la fecha y los 3 valores recolectados (por ejemplo 202109241245;0.18;2.5;1348).

La named FIFO de salida se llamará stats_loadavg

2. Deberá crear otro programa "colector" que toma los datos que dejó el primer programa, los organice por fecha guardándolos en un archivo por día. El resultado será un archivo a guardar en disco con el nombre: stats_sistema_AAAAMMDD.txt

3. Deberá crear un tercer programa que enviará los archivos creados (stats_sistema_*) a las IPs descriptas en el archivo "`sistemas_ips.txt`" (una IP por línea), este programa podrá manejar tantos hijos como la máquina soporte (máximo un hijo por núcleo del procesador). En el archivo `cpuinfo.txt` tendrá un número (en ASCII) que representa la cantidad de núcleos del procesador que tiene el sistema.

El programa actuará como cliente conectandose a las IPs server de la lista en "`sistemas_ips.txt`" usando el puerto 3000 vía TCP:

1. Enviará primero un long con el tamaño del archivo.
2. Luego enviará el contenido completo del archivo.
3. Al finalizar se terminará la ejecución del hijo (de manera ordenada).

Esto sucederá para cada IP del archivo hasta finalizar la lista y sin superar la cantidad de procesos simultáneos del sistema. Para resolver este ejercicio puede usar la librería de sockets de la cátedra que se presenta en la siguiente página.

| Apellido y Nombres | Legajo | Profesor | # Hojas |
|---|---|---|---|
| | | | |

Función de librería de TCP/IP de la cátedra:

```
#define PORT 3490   /* El puerto donde se conectará, servidor */
#define BACKLOG 10  /* Tamaño de la cola de conexiones recibidas */

/*
  Función cliente que se conecta a un servidor remoto usando TCP
   - Toma: Los parámetros pasados por línea de comandos
           Por ejemplo: micliente 127.0.0.1 3490
   - Devuelve: El socket creado listo para ser usado (si es !=0)
*/

int    conectar (int argc, char **argv);

/* Crea un socket servidor y lo devuelve
    - Toma: La estructura con los datos del socket a configurar ya armada
    - Devuelve: El socket creado para aceptar conexiones (si es != 0)
*/

int    Open_conection (struct sockaddr_in *);

/* Función que acepta una conexión entrante (bloqueante)
    - Toma: El socket creado por conectar()
    - Devuelve: Un socket ya conectado a un cliente.
*/

int    Aceptar_pedidos (int socket);
```