	Universidad Tecnológica Nacional **Facultad Regional Buenos Aires** **Departamento de Ingeniería Electrónica**	**Informática I** 27 de Noviembre de 2021 2do parcial – Tema 1

Apellido y Nombres	Legajo	# de Hojas

Normas Generales

Numere las hojas entregadas. **Lea detenidamente cada pregunta y consulte las dudas de interpretación que pudiesen surgir.** Complete en la primera hoja la cantidad total de hojas entregadas.
Realice este parcial en lápiz, o con tinta color azul o negro. ***Por favor NO utilice rojo ni verde***.
Identificar cada hoja con: Nombre, Apellido, Legajo.
Por favor entregar esta hoja y las restantes del tema junto al examen.

Parte Teórica

1) ¿Cuales 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?
2) ¿Que haría para realizar una tarea antes de que el programa se cierre cuando se recibe un "kill"?
3) Describa los pasos que se deben realizar para crear un socket servidor TCP/IP y los pasos a realizar para crear un socket TCP/IP cliente.

Parte Práctica

Escribir las siguientes partes de un programa que transfiere archivos vía TCP/IP

1) Escribir una función llamada load_conf() que tome un nombre de archivo (como cadena), lo abra y devuelva una estructura con los campos port, backLog y archStr que toma del archivo de configuración. El archivo tendrá el siguiente formato:

```
PORT=num
BACKLOG=num
ARCHIVOS=arch01.txt,arch02.txt,…
```

   No se sabe cuantos archivos vienen, pero a los fines de esta función es una cadena a leer que tienen como máximo 1024 caracteres. Para este ejercicio no puede utilizar strtok(), realicé la separación dentro de la función.
2) Crear una función que reciba un listado de strings separados por comas y devuelva una estructura con un puntero a un vector de punteros a string y un entero indicando la cantidad de strings (archivos) que tiene el vector. El vector (creado por esta función) debe contener una puntero a cada string de la cadena. Para la realización de esta función puede utilizar strtok().
3) Realizar una función TransferFilesData() que recibirá un entero representando un socket, un vector de strings y un entero con su tamaño. La función deberá abrir cada archivo dentro del vector y transferir su contenido. Una vez transferidos todos los archivos finalizará.
4) Crear el programa que lea la configuración (usando la función creada en 1) desde un archivo pasado como primer argumento del main. Luego llamar a la función initSocketS() (ya creada) que toma el port (provisto por la configuración) inicializando el socket a utilizar en este servidor. La función initSocketS() devuelve un entero que es el número de socket listo para ser utilizado en un accept(). Esta función se deberá quedar esperando hasta backLog conexiones. Cada vez que reciba una nueva conexión deberá crear un proceso nueva para manejar la transferencia. Realice además dentro del main todos los pasos que considere necesarios para finalizar el programa liberando todos los recursos utilizados.
5) Crear un programa cliente que llame a la función initSocketC() que no toma nada (pide la IP y puerto sola) y devuelve un socket ya abierto que se deberá utilizar para recibir datos. Estos datos recibidos se deberan guardar en el archivo recibido_pid.txt (deberá conseguir el process ID del cliente para esto).

Nota: Si desea en su lugar utilizar la librería de la cátedra, se adjunta en la siguiente sección sus encabezados para su comodidad.

	Universidad Tecnológica Nacional **Facultad Regional Buenos Aires** **Departamento de Ingeniería Electrónica**	**Informática I** 27 de Noviembre de 2021 2do parcial – Tema 1

Apellido y Nombres	Legajo	# de 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);
```