FINAL INFORMATICA I	Duración	Fecha	Hojas
	19:05 a 20:45	21/may/2025	

Nombre y Apellido	Nº Legajo	Calificación		Docente Evaluador	
		número	letras	Nombre	Firma

Numere las hojas entregadas, y complete el casillero ***Hojas*** *con la cantidad.*
Lea detenidamente el enunciado, su correcta interpretación forma parte de esta evaluación.

El uso de variables globales está restringido a los casos que sea de uso sea mandatorio (es decir si no se pueda resolver mediante pasaje de datos por parámetros).

Requerimiento:

Se debe realizar una aplicación servidora que funcione en modo concurrente (es decir que pueda atender a varios clientes en forma simultánea), la cual debe estar diseñada para responder la cantidad de bytes que posee un string recibido.

Al iniciar, la aplicación servidora habilitará un puerto TCP/IP en donde esperará a las consultas de los diferentes clientes. El puerto a utilizar esta definido por la etiqueta PORT_NUMBER.

Ante una solicitud de conexión, la aplicación servidora creará un hilo que atenderá las consultas de este nuevo cliente. En caso de error al generar un nuevo hilo o solicitar memoria dinámica, se debe emitir un mensaje de error por el stderr y continuar.

La aplicación finaliza al recibir la señal SIGUSR1.

Los hilos dedicados a atender a un cliente, quedarán esperando la recepción de un string (que finaliza con '\0'), al recibir un texto deben obtener la cantidad de caracteres del texto (omitiendo el '\0') y responder dicha cantidad (en modo texto). Largo máximo de un string 1020 bytes.

Cuando un hilo recibe un texto que inicia con "FIN", debe cerrar la conexión con el cliente (sin responder la cantidad) y finalizar el hilo. Lo mismo si se detecta el cierre de la conexión por parte del cliente.

Listado de funciones útiles

La siguiente es una lista de funciones que les puede ser de utilidad.

Bajo ningún aspecto debe considerar ni que debe usar todas las funciones de esta lista, ni que no puede utilizar otras.

Funciones de la cátedra para manejo de sockets: (network.c y .h)

```
int OpenServer(int portnr);
```

vincula al proceso (server) con un puerto de la computadora vía SO

parámetro de entrada

`portnr`: Numero de puerto en el que se busca dar disponibilidad el servicio

valor de retorno:

-1: error en el proceso

En caso de éxito, un descriptor de archivo para el nuevo socket es devuelto

```
int CloseServer(int socketfd);
```

Cierra el socket y libera el recurso

parámetro de entrada

`socketfd`: Valor obtenido mediante OpenServer (si este fue >0)

```
int WaitConnect (int msSock);
```

Establece una nueva conexión con un cliente

parámetro de entrada

`msSock`: Valor de socket obtenido mediante OpenServer (si este fue >0)

valor de retorno:

-1: error en el proceso

En caso de éxito retorna un valor mayor a cero, que podrá ser utilizado por recv/send

```
int CloseConnect(int sockfd);
```

Cierra la conexión y libera el recurso

parámetro de entrada

`sockfd`: Valor de socket obtenido mediante WaitConnect (si este fue >0)

Hilos:

```
int pthread_create( pthread_t * thread,const pthread_attr_t * attr,
                    void *(*start_routine)(void *), void * arg);
int pthread_join (pthread_t thread, void **retval);
int pthread_detached (pthread_t thread);
int pthread_detach(pthread_t thread);
pthread_t pthread_self(void);
int pthread_exit (void *retval);
int pthread_mutex_init(pthread_mutex_t * mutex, const pthread_mutexattr_t *attr);
int pthread_mutex_destroy(pthread_mutex_t *mutex);
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

señales:

```
sighandler_t signal(int signum, sighandler_t handler);
```

Memoria dinámica:

```
void *malloc(size_t size);
void free(void *ptr);
```

Lectura / escritura de sockets:

```
ssize_t send(int sockfd, const void *buf, size_t len, int flags);
ssize_t recv(int sockfd, void *buf, size_t len, int flags);
tembien puede utilizar read y write
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
```

Funciones varias:

```
int strncmp(const char *s1, const char *s2, size_t n);
int strcmp(const char *s1, const char *s2);
int strlen (const char *str);
```