Objetivos

Objetivo general

Resolver un problema de concurrencia empleando las **primitivas POSIX** específicas de gestión de semáforos y memoria compartida.

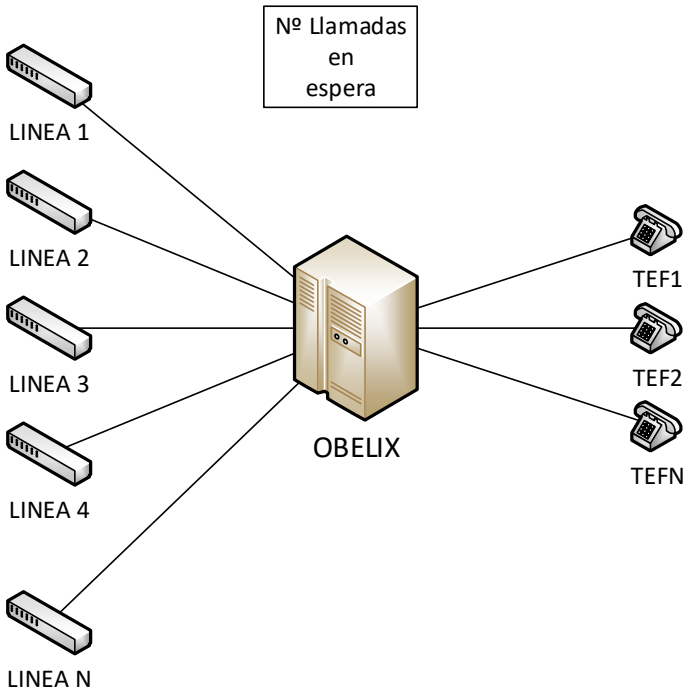
Objetivos específicos

El alumno deberá adquirir las habilidades para la resolución de un problema concurrente utilizando los mecanismos de sincronización y comunicación entre procesos basados en memoria compartida y semáforos.

Implementar la solución teórica obtenida mediante los mecanismos que proporciona el estándar POSIX. Estos son los llamados IPC y más concretamente la implementación de segmentos de memoria compartida y semáforos nombrados. El alumno deberá adquirir las habilidades necesarias para la creación y espera de procesos así como el correcto tratamiento de las señales básicas relativas a la finalización de la aplicación.

Especificación

Es nuestra intención crear un sistema básico de control de llamadas telefónicas denominado **Obelix**. En este sistema, nos encontraremos con un proceso Manager que será el encargado del control de **Obelix**. Tendremos, por un lado, *líneas* que recibirán llamadas telefónicas, y por otro lado, tendremos *teléfonos* que serán los encargados de atender estas llamadas. Además, nos piden que en todo momento podamos informar de las llamadas que, habiendo entrado a las líneas, se encuentren *en espera* de poder ser atendidas por los teléfonos.
Este sería el esquema de Obelix:



Construya un sistema compuesto por tres ejecutables que simule el funcionamiento que se detalla a continuación, y que es, la especificación básica de funcionamiento del sistema **Obelix**.

- ***Manager***: Deberá cumplir las siguientes especificaciones:
    - Recibirá dos argumentos: <nº teléfonos> <nº líneas>
    - Inicializará los semáforos y la memoria compartida para almacenar el número de llamadas en espera.
    - Lanzará el número de procesos *teléfono* y procesos *línea* especificado en los argumentos.
    - Realizará el apagado controlado de **Obelix**. Para ello, esperará la finalización automática de todos los procesos *línea* y forzará la finalización de los procesos *teléfono* una vez hayan finalizados todos los procesos *línea*.
    - Controlará el *Apagado de emergencia* de **Obelix**, ante la pulsación de Ctrl-C. Esto es:
        - Forzará la finalización de todos los procesos *línea* actualmente en funcionamiento.
        - Forzará la finalización de todos los procesos *teléfono* existentes.
    - Por último, pero no menos importante, liberará todos los recursos utilizados. (semáforos, memoria compartida, etc.)

- ***Teléfono:*** Realizará las siguientes tareas dentro de su función:
    - Se inician en espera de una llamada, informando de ello: “Teléfono[UID] en espera…”
    - Cuando reciben una llamada, tienen una conversación variable entre 10..20 segundos e informan de ello: “Teléfono[UID] en conversación…”, además en esa misma línea informarán del número de llamadas en espera en ese momento:"Nº de llamadas en espera: [shm]” y decrementarán el número de llamadas en espera.
    - Cuando finalicen su llamada, volverán al estado de espera inicial.
    - *En ningún caso finalizarán su ejecución salvo cuando el proceso Manager lo considere oportuno.*

- ***Línea:*** Realizará las siguientes tareas dentro de su función:
    - Se inician y esperan una llamada (Simulado con un rand() entre 1..30 segundos), informando de ello: “Linea[UID] esperando llamada”.
    - Cuando reciben la llamada (Finalización de rand()):
        - Incrementarán el contador de llamadas en espera.
        - Esperarán a que exista un teléfono libre, informando de ello: “Linea[UID] esperando teléfono libre…”, además en esa misma línea informarán del número de llamadas en espera en ese momento:"Nº de llamadas en espera: [shm]”.
        - Cuando exista un teléfono libre, desviarán la llamada hacia el teléfono, informando de ello: “Linea[UID] desviando llamada a un teléfono…” y finalizarán su trabajo

Ejemplo de ejecución

Una vez resuelto el ejercicio, si se ejecuta el proceso *manager* con los siguientes argumentos (*make test*),

```
$ exec/manager 3 10
```

el resultado debe ser similar al siguiente (cambiará el PID de los procesos, el orden de impresión y los valores generados de manera aleatoria):

```
[MANAGER] 3 teléfonos creados.
Teléfono [38106] en espera...
Teléfono [38107] en espera...
Teléfono [38108] en espera...
[MANAGER] 10 lineas creadas.
Linea [38114] esperando llamada...
Linea [38112] esperando llamada...
Linea [38115] esperando llamada...
Linea [38111] esperando llamada...
Linea [38118] esperando llamada...
Linea [38116] esperando llamada...
Linea [38117] esperando llamada...
Linea [38109] esperando llamada...
Linea [38113] esperando llamada...
Linea [38110] esperando llamada...
Linea [38117] esperando telefono libre...Nº Llamadas en espera: 1
Linea [38117] desviando llamada a un telefono...
Teléfono [38106] en conversacion... Nº Llamadas en espera: 0
[MANAGER] Proceso LINEA terminado [38117]...
Linea [38114] esperando telefono libre...Nº Llamadas en espera: 1
Linea [38114] desviando llamada a un telefono...
Teléfono [38107] en conversacion... Nº Llamadas en espera: 0
[MANAGER] Proceso LINEA terminado [38114]...
Linea [38118] esperando telefono libre...Nº Llamadas en espera: 1
Linea [38118] desviando llamada a un telefono...
Teléfono [38108] en conversacion... Nº Llamadas en espera: 0
[MANAGER] Proceso LINEA terminado [38118]...
Linea [38111] esperando telefono libre...Nº Llamadas en espera: 1
Linea [38116] esperando telefono libre...Nº Llamadas en espera: 2
Teléfono [38107] en espera...
Linea [38111] desviando llamada a un telefono...
Teléfono [38107] en conversacion... Nº Llamadas en espera: 1
[MANAGER] Proceso LINEA terminado [38111]...
Teléfono [38106] en espera...
Linea [38116] desviando llamada a un telefono...
Teléfono [38106] en conversacion... Nº Llamadas en espera: 0
[MANAGER] Proceso LINEA terminado [38116]...
Linea [38115] esperando telefono libre...Nº Llamadas en espera: 1
Teléfono [38108] en espera...
Linea [38115] desviando llamada a un telefono...
Teléfono [38108] en conversacion... Nº Llamadas en espera: 0
[MANAGER] Proceso LINEA terminado [38115]...
Linea [38110] esperando telefono libre...Nº Llamadas en espera: 1
Linea [38113] esperando telefono libre...Nº Llamadas en espera: 2
Linea [38112] esperando telefono libre...Nº Llamadas en espera: 3
Linea [38109] esperando telefono libre...Nº Llamadas en espera: 4
Teléfono [38106] en espera...
Linea [38110] desviando llamada a un telefono...
Teléfono [38106] en conversacion... Nº Llamadas en espera: 3
[MANAGER] Proceso LINEA terminado [38110]...
Teléfono [38107] en espera...
Linea [38113] desviando llamada a un telefono...
Teléfono [38107] en conversacion... Nº Llamadas en espera: 2
[MANAGER] Proceso LINEA terminado [38113]...
Teléfono [38108] en espera...
Linea [38112] desviando llamada a un telefono...
Teléfono [38108] en conversacion... Nº Llamadas en espera: 1
[MANAGER] Proceso LINEA terminado [38112]...
Teléfono [38106] en espera...
Linea [38109] desviando llamada a un telefono...
Teléfono [38106] en conversacion... Nº Llamadas en espera: 0
[MANAGER] Proceso LINEA terminado [38109]...

----- [MANAGER] Terminar con cualquier proceso pendiente ejecutándose ----
-
[MANAGER] Terminando proceso TELEFONO [38106]...
[MANAGER] Terminando proceso TELEFONO [38107]...
[MANAGER] Terminando proceso TELEFONO [38108]...

[MANAGER] Terminacion del programa (todos los procesos terminados).
```

Sugerencias

Necesitaremos, al menos dos semáforos para controlar la interacción entre líneas y teléfonos.
Debemos pensar muy bien como gestionamos, desde el proceso *manager,* los procesos que debemos esperar que finalicen y los procesos que debemos finalizar nosotros.
Ojo con la memoria compartida, no entremos en condición de carrera. Con esto se os da otra pista más.

Bibliografía básica

- Kernighan B. W., Ritchie D. M. - *El lenguaje de programación C* - Ed. Prentice Hall 1991.
- Páginas del manual electrónico de UNIX.
- Rochkind, M.J. - *Advanced Unix Programming (2 nd Edition)* - Ed. Addison-Wesley 2004.
- Robbins, K.A., Robbins, S – *UNIX Programación Práctica* – Ed. Prentice Hall 1997.

Plazos de entrega

Esta práctica deberá:

- **Entregarse como máximo el día 18 de abril de 2023 a las 23:59 horas.**
- No se tendrán en cuenta las prácticas entregadas fuera de plazo.
- No se admitirán prácticas entregadas por email.
- La entrega de la práctica tendrá que realizarse en formato electrónico.
- La entrega será mediante una tarea en el espacio virtual de la asignatura en *Campus Virtual*. Si algún alumno no está registrado en dicha plataforma, deberá informar al profesor de prácticas para que le añada al curso.
- El fichero comprimido subido al sistema deberá cumplir la especificación que se indica en la sección Normas de Presentación.

Normas de presentación

El seguimiento de las normas de presentación de la práctica es **obligatorio**, especialmente la parte relativa a la compilación automática mediante *make* de todos los ejecutables necesarios para la resolución de la misma. Obviamente, las rutas del fichero *makefile* deberán ser relativas para permitir su compilación en diferentes computadores sin necesidad de realizar ningún cambio. En este curso únicamente será posible entregar la práctica mediante *Campus Virtual*, es decir, **no hay que realizar ninguna impresión** de la misma.

Únicamente se subirá un fichero comprimido en formato zip por un solo miembro de la pareja. Debe cumplir el siguiente patrón de nombrado:

primerApellido1_segundoApellido1_alumno1_primerApellido2_segundoApellido2_alumno2.zip

Al descomprimir el fichero, se deberá obtener la siguiente estructura de directorios:

/include	Contendrá los ficheros de cabecera .h
/src	Contendrá los ficheros fuente .c
/obj	Contendrá los ficheros objeto (se deberán crear mediante make).
/exec	Contendrá los ejecutables (se deberán crear mediante make).
/data	Contendrá los ficheros para ejecutar las pruebas.
/doc	Contendrá un fichero .pdf con la memoria de la práctica.
/	El directorio raíz contendrá únicamente dos ficheros: • Un archivo makefile que deberá permitir compilar todos los programas relacionados con la práctica mediante la llamada directa a make. • Un fichero ascii (author.txt) que contendrá nombre y apellidos de los autores.

La **memoria** de la práctica debe incluir una discusión de la solución planteada, haciendo hincapié en aquellas partes del código fuente que considere más relevante. Así mismo, la memoria ha de incluir una sección en la que se indique brevemente cómo compilar y ejecutar.

Evaluación

Se valorarán dos aspectos: funcionalidad y calidad. La funcionalidad valora que se resuelva el problema conforme a sus especificaciones y que sea consistente, es decir, funcione adecuadamente en todas y cada una de las ejecuciones. Si solo funciona a veces, se considera que no funciona. Si existen casos no contemplados o faltan mecanismos de gestión de errores, la puntuación se verá afectada negativamente, aunque el programa nunca falle. En cuanto a la calidad del código, se valoran los siguientes aspectos:

- Estructura y organización del código, mediante clases, funciones o métodos.
- Elección de nombres significativos para funciones, variables, etc.
- Evitar uso de variables globales innecesarias.
- Evitar anidación injustificada de estructuras de control.
- Evitar uso incorrecto de tipos de datos, código redundante, código muerto, etc.
- Evitar bloques (funciones/métodos) demasiado largos.

La **valoración de la calidad es negativa**, es decir, restará sobre la nota final obtenida en funcionalidad. Esto implica que una práctica completamente funcional puede no obtener la máxima nota. Es importante destacar que el alumno **debe ser capaz de defender su trabajo**, por lo que en cualquier momento se le puede **solicitar que defienda el trabajo** presentado.

Penalizaciones

Se contemplan 2 motivos por los que la puntuación total de la actividad puede sufrir penalizaciones:

- Errores de sintaxis que impiden ejecutar el programa.
- Incumplir otras condiciones del enunciado.

En el caso de errores de sintaxis, si supone cambios menores, se pedirá al alumno presentar una versión corregida.

Condiciones

El alumno debe tener especial cuidado con la **incorporación de código ajeno** en su programa. Está PERMITIDO siempre que se cumplan las siguientes condiciones:

- Los fragmentos copiados y su autor están claramente identificados.
- El alumno comprende y es capaz de explicar con TODO DETALLE la funcionalidad del código copiado.
- El autor del código original permite la copia mediante la licencia correspondiente y esta aparece claramente junto al código original.
- El autor del original mantiene su código públicamente accesible (y el alumno proporciona el enlace en forma de comentario).
- El autor del original no es ni ha sido alumno de la UCLM.

Infringir cualquiera de estas normas al utilizar código ajeno se considerará plagio. Cometer plagio implica una calificación de 0 en la actividad *«Realización de prácticas de laboratorio»* según el artículo 9 de la *«Guía de Evaluación del Estudiante»*:

> *Art. 9. Realización fraudulenta de pruebas de evaluación.*
> *1. La constatación de la realización fraudulenta de una prueba de evaluación o el incumplimiento de las instrucciones fijadas para la realización de la prueba dará lugar a la calificación de suspenso (con calificación numérica de 0) en dicha prueba. En el caso particular de las pruebas finales, el suspenso se extenderá a la convocatoria correspondiente.*