## Sistemas Operativos – Grado en Ingeniería Informática

## Práctica obligatoria 2

# Programación con threads

## Descripción

El objetivo de la práctica es que el alumno se familiarice con la sincronización de procesos y *threads*. Para ello, se va a plantear un problema en el que se simulará una campaña de vacunación ante una pandemia usando procesos ligeros o *threads*.

En concreto, se desea plantear un escenario para un país pequeño, donde cada habitante va a intentar ser vacunado acudiendo a uno de los 5 centros de vacunación de referencia que hay en su territorio. Tres empresas farmacéuticas se encargan de la fabricación de vacunas, sabiendo que en cada periodo solo podrán fabricar unas pocas. La aplicación gestionará mediante *threads* el funcionamiento de la campaña de vacunación. Por limitaciones de tiempo y espacio, para la implementación se utilizará un número reducido de habitantes y vacunas de acuerdo con lo siguiente:

- Cada centro de vacunación comenzará teniendo un cierto número de vacunas disponibles, al margen de las que cada empresa fabricará después.
- Cada empresa fabricante tiene encargado un cierto número de vacunas que debe entregar. El número total de vacunas fabricadas en total por las 3 empresas será igual al número de habitantes del país para que todo el mundo pueda vacunarse.
- A la larga, las 3 empresas fabricarán el mismo número de vacunas. Sin embargo, no tienen capacidad de fabricarlas todas a la vez, por lo que tardan un tiempo aleatorio en producir un número pequeño de vacunas (también aleatorio) que repartirán entre los 5 centros de vacunación antes de fabricar más.
- El número total de vacunas fabricadas en total por las 3 empresas será igual al número de habitantes del país para que todo el mundo pueda vacunarse.
- El reparto de las vacunas entre los centros de vacunación no es inmediato, sino que cada empresa tarda un tiempo aleatorio en realizarlo. La cantidad de vacunas que cada fábrica entrega a cada centro en cada tanda de vacunas debería depender de la demanda que hay en cada uno de ellos.
- Cada vez que una fábrica realiza el reparto de una tanda de vacunas a un centro de vacunación se debe mostrar por pantalla el número de vacunas que le entrega.
- Los habitantes son avisados para acudir a vacunarse por tandas, de forma que no se llama a más personas hasta que toda la anterior tanda ha sido vacunada. Este comportamiento se repetirá de forma continua hasta que todas las personas hayan sido vacunadas. Las autoridades sanitarias del país han establecido que habrá 10 tandas de vacunación para inmunizar a la población completa.
- Cuando una persona recibe la cita de vacunación selecciona un centro según su propio interés (podría hacerlo aleatoriamente). Si cuando llega al centro no hay vacunas disponibles en el mismo, deberá esperar allí hasta que lleguen más.

- Todos los números aleatorios indicados en los puntos anteriores podrán calcularse utilizando la función rand().

Se utilizará un fichero de texto como entrada para saber el número total de habitantes, cuántas vacunas iniciales habrá en cada centro, cuántas vacunas podrá producir cada fábrica en una tanda y el tiempo de fabricación de la misma y el tiempo de reparto de la tanda entre los centros, más el tiempo que tarda cada persona en darse cuenta de que ha recibido la cita y el que tarda en desplazarse al centro de vacunación. Este fichero tendrá únicamente un número entero por línea en el siguiente orden:

- Habitantes totales (por defecto 1200)
- Vacunas iniciales por centro (por defecto 15)
- Número mínimo de vacunas producidas por una fábrica en una tanda (por defecto 25)
- Número máximo de vacunas producidas por una fábrica en una tanda (por defecto 50)
- Tiempo mínimo de fabricación de una tanda en una fábrica (por defecto 20)
- Tiempo máximo de fabricación de una tanda en una fábrica (por defecto 40)
- Tiempo máximo de reparto de una tanda de vacunas a un centro de vacunación (por defecto 3). El tiempo mínimo de reparto siempre es 1.
- Tiempo máximo que tarda un habitante hasta que se da cuenta que le han citado para vacunarse (por defecto 4). El tiempo mínimo de reacción del habitante a la cita es siempre 1.
- Tiempo máximo de desplazamiento de un habitante al centro de vacunación (por defecto 2). El tiempo mínimo de desplazamiento siempre es 1.

El nombre del fichero de entrada se pasará como argumento del programa.

La salida del programa debería ser algo del tipo:

```
VACUNACIÓN EN PANDEMIA: CONFIGURACIÓN INICIAL
Habitantes: 1200
Centros de vacunación: 5
Fábricas: 3
Vacunados por tanda: 120
Vacunas iniciales en cada centro: 15
Vacunas totales por fábrica: 400
Mínimo número de vacunas fabricadas en cada tanda: 25
Máximo número de vacunas fabricadas en cada tanda: 50
Tiempo mínimo de fabricación de una tanda de vacunas: 20
Tiempo máximo de fabricación de una tanda de vacunas: 40
Tiempo máximo de reparto de vacunas a los centros: 3
Tiempo máximo que un habitante tarda en ver que está citado para vacunarse: 4
Tiempo máximo de desplazamiento del habitante al centro de vacunación: 2

PROCESO DE VACUNACIÓN
Habitante 1 elige el centro 1 para vacunarse
Habitante 3 elige el centro 3 para vacunarse
Habitante 7 elige el centro 2 para vacunarse
Habitante 12 elige el centro 5 para vacunarse
                ...
Habitante 12 vacunado en el centro 5
Habitante 1 vacunado en el centro 1
Habitante 3 vacunado en el centro 3
```

...
Habitante 97 elige el centro 2 para vacunarse
Habitante 99 elige el centro 2 para vacunarse
Habitante 65 elige el centro 1 para vacunarse
Habitante 100 elige el centro 5 para vacunarse
Habitante 64 elige el centro 1 para vacunarse
Fábrica 2 prepara 47 vacunas
Fábrica 2 entrega 6 vacunas en el centro 1
Fábrica 2 entrega 11 vacunas en el centro 2
Fábrica 2 entrega 12 vacunas en el centro 3
Fábrica 2 entrega 9 vacunas en el centro 4
Fábrica 2 entrega 9 vacunas en el centro 5
Habitante 99 vacunado en el centro 2
Habitante 65 vacunado en el centro 1
...
Habitante 51 vacunado en el centro 3
Habitante 35 vacunado en el centro 3
Fábrica 1 prepara 50 vacunas
Fábrica 1 entrega 11 vacunas en el centro 1
Fábrica 1 entrega 13 vacunas en el centro 2
Fábrica 1 entrega 5 vacunas en el centro 3
Fábrica 1 entrega 11 vacunas en el centro 4
Fábrica 1 entrega 10 vacunas en el centro 5
Habitante 61 vacunado en el centro 1
...
Habitante 175 vacunado en el centro 2
Fábrica 3 prepara 32 vacunas
Fábrica 3 entrega 8 vacunas en el centro 1
Fábrica 3 entrega 10 vacunas en el centro 2
Fábrica 3 entrega 7 vacunas en el centro 3
Fábrica 3 entrega 4 vacunas en el centro 4
Fábrica 3 entrega 3 vacunas en el centro 5
Habitante 137 vacunado en el centro 4
Habitante 195 vacunado en el centro 1
...
Habitante 1150 vacunado en el centro 1
Fábrica 1 prepara 31 vacunas
Fábrica 1 entrega 6 vacunas en el centro 1
Fábrica 1 entrega 8 vacunas en el centro 2
Fábrica 1 entrega 4 vacunas en el centro 3
Fábrica 1 entrega 7 vacunas en el centro 4
Fábrica 1 entrega 6 vacunas en el centro 5
Fábrica 1 ha fabricado todas sus vacunas
Habitante 1179 vacunado en el centro 4
...
Habitante 1181 vacunado en el centro 4
Habitante 1199 vacunado en el centro 3
Habitante 1149 vacunado en el centro 1
Vacunación finalizada

La salida se presentará por pantalla y también se almacenará en un fichero cuyo nombre se pasará por argumento al programa.

## Objetivos parciales

- Gestionar correctamente la fabricación de vacunas. Cada fábrica tiene asignado un número de vacunas que debe fabricar, y las va fabricando por tandas. Cada tanda se reparte entre los 5 centros de vacunación de manera que todos recibirán a la larga vacunas suficientes para poder atender a todas las personas que acudan al mismo. No se deben producir interbloqueos ni inanición (**hasta 3 puntos**).
- Emular el comportamiento de los habitantes. Cuando a un habitante le corresponde vacunarse, elige un centro de vacunación de su interés (podría hacerlo aleatoriamente), se desplaza al centro y se vacuna si hay vacunas disponibles, y si no espera a vacunarse en el centro hasta que las haya. No se deben producir interbloqueos ni inanición (**hasta 3 puntos**).
- Leer la configuración inicial del problema (habitantes, horquillas de tiempo, etc) desde un fichero de texto (**hasta 1 punto**).
- Escribir por pantalla cómo se desarrolla el proceso de vacunación indicando la configuración inicial del problema, la fabricación y reparto de vacunas entre los centros de vacunación por parte de las fábricas, y la elección de centro por parte de los habitantes y su posterior vacunación (**hasta 0,5 puntos**).
- Escribir la información anterior en un fichero de texto (**hasta 0,5 puntos**).
- Pasar por argumentos los nombres de los ficheros de entrada y de salida. En caso de que no se introduzca el fichero de salida, su nombre por defecto será **salida_vacunacion.txt**. Si no se pasa el nombre del fichero de entrada, su nombre por defecto será **entrada_vacunacion.txt**. Es decir, el programa tiene que entender (**hasta 1 punto**):

```
vacunacion
vacunacion nombre_fichero_entrada
vacunacion nombre_fichero_entrada nombre_fichero_salida
```

- Escribir en pantalla la estadística final del proceso de vacunación. Por cada fábrica se indicará cuántas vacunas ha fabricado y cuántas ha entregado a cada centro de vacunación. Por cada centro de vacunación se indicará cuántas vacunas ha recibido en total, cuántos habitantes se han vacunado en él y cuántas vacunas le han sobrado al final del proceso (**hasta 0,5 puntos**).
- Escribir en el fichero de salida la estadística final del proceso de vacunación en los términos indicados en el punto anterior (**hasta 0,5 puntos**).

**Nota:** Las puntuaciones para cada objetivo parcial son las puntuaciones máximas que se pueden obtener si se cumplen esos objetivos.

**Nota:** No se debe hacer un programa separado para cada objetivo, sino un único programa genérico que cumpla con todos los objetivos simultáneamente.

**Nota:** Para evitar que se produzcan interbloqueos o inanición, no se considerarán válidas soluciones a medida implementadas por el alumno que solucionen casos concretos.

**Nota:** No se puede crear simultáneamente un número cualquiera de *threads* en el sistema. El número máximo de *threads* simultáneos que pueden crearse depende, entre otras cosas, del tamaño de memoria principal y del tamaño de página del sistema. Por esto los *threads* que modelan el comportamiento de los habitantes deben crearse por tandas.

## Entrega de prácticas

La entrega se realizará a través del Campus Virtual en las fechas anunciadas en el mismo. Se debe entregar un único archivo *vacunacion.c* con el código de toda la práctica, debidamente comentado y una memoria de la misma en formato PDF. La memoria debe incluir:

- Índice de contenidos
- Autores
- Descripción de la información sobre la concurrencia: se indicarán cuáles son los threads existentes en el programa, los recursos que utilizarán de forma compartida, las condiciones de sincronización en el acceso a los mismos y los mecanismos de comunicación y sincronización empleados en la solución.
- Descripción de las variables globales utilizadas, destacando cuáles constituyen los recursos compartidos del programa:

| Variables globales | Tipo | Descripción |
|---|---|---|
| | | |
| | | |
| | | |

- Descripción de las principales funciones implementadas, incluyendo la función principal *main*. En el caso de los threads, describir la sección crítica elegida.

| | Nombre de la función | Nombre | Tipo | Descripción |
|---|---|---|---|---|
| **Argumentos** | **Argumento 1** | | | |
| | **Argumento 2** | | | |
| | **… … … …** | | | |
| **Variables Locales** | **Variable 1** | | | |
| | **Variable 2** | | | |
| | **Variable 3** | | | |
| | **… … … …** | | | |
| **Valor devuelto** | | | | |
| **Descripción de la función** | | | | |

- Comentarios personales: incluyendo problemas encontrados, crítica constructiva, propuesta de mejoras y evaluación del tiempo dedicado.
- No incluir código fuente (salvo el estrictamente necesario para complementar las descripciones anteriores).
- NO DESCUIDE LA CALIDAD DE LA MEMORIA DE SU PRÁCTICA. Aprobar la memoria es tan imprescindible para aprobar la práctica, como el correcto funcionamiento de la misma. Si al evaluar la memoria se considera que no alcanza el mínimo admisible, la práctica se considerará SUSPENSA.

## Normas de estilo en el código fuente

El código fuente se evaluará por su simplicidad, la presencia de comentarios, la óptima gestión de recursos y la gestión de errores.

A la hora de codificar la solución pedida, se deberán respetar una serie de normas de estilo:

- Las variables locales de las funciones se declararán inmediatamente después de la llave de comienzo del cuerpo de la misma. Se penalizará la declaración de variables entre sentencias ejecutables de las funciones.
- No se admitirán asignaciones iniciales a variables cuyo valor dependa del valor de otras variables. El valor asignado en estos casos siempre deberá ser conocido en tiempo de compilación.
- Cuando se declare una variable de tipo *array*, su tamaño deberá ser conocido en tiempo de compilación. Si se quiere utilizar un *array* de tamaño variable, deberá crearse en memoria dinámica mediante las funciones correspondientes (malloc, calloc o realloc).
- Las operaciones sobre *strings* (copia, comparación, duplicación, concatenación, etc) se realizarán en lo posible mediante las funciones indicadas en *string.h* (ver referencias a la biblioteca de C en el Campus Virtual).
- En general, se penalizará el uso de construcciones propias de C++.
- Al compilar el código fuente, deberá producirse el menor número posible de *warnings* (mejor que no se produzca ninguno).

El incumplimiento de estas normas de estilo, así como de otras normas que puedan ser anunciadas por el profesor a través del Campus Virtual, conllevará una penalización en la nota obtenida.

## Defensa oral de la práctica

Después de entregar la práctica, cada alumno será convocado de forma **individual** para realizar una defensa oral de su práctica, en fecha y lugar indicados por el equipo docente de la asignatura.

En la defensa oral de la práctica, el alumno:

- Compilará y montará el ejecutable de su práctica desde la línea de mandatos de Linux.
- Ejecutará y demostrará el funcionamiento de su práctica, de acuerdo con las indicaciones del profesor.
- Responderá a cuestiones relativas al código fuente de su práctica.

La defensa oral de la práctica tiene influencia sobre la nota final obtenida en la misma, de forma que puede mejorar o empeorar la nota parcial calculada a partir del cumplimiento de objetivos, las normas de estilo del código fuente y la memoria. Si el desempeño del alumno en la defensa oral no alcanza el mínimo nivel admisible, la práctica en su conjunto se considerará SUSPENSA, pudiéndose incluso solicitar al alumno que entregue una práctica totalmente nueva para la siguiente convocatoria.

## Evaluación de la práctica

La nota de la práctica será individual para cada alumno, y se calculará a partir de:

- El cumplimiento de los objetivos planteados en la misma.
- La aplicación de las normas de estilo.
- La memoria.
- La defensa oral individual.

## Autoría de la práctica

La práctica se debe realizar en grupos de 2 personas como máximo.

El hecho de detectar copia en las prácticas expondrá a los alumnos a la posibilidad de una apertura de expediente disciplinario y expulsión. En caso de detectar copia, los alumnos afectados serán suspendidos en la TOTALIDAD de la asignatura. Una práctica será considerada copia en caso de que contenga la totalidad o una parte de la práctica de otro alumno. Se considerará copia en caso de:

- Archivos que contengan la totalidad o fragmentos de código de otro alumno
- Memorias con la totalidad o fragmentos de frases e imágenes de otro alumno

El profesor podrá hacer uso de detectores automáticos de plagio en las prácticas, tanto en la parte referente al código como a la memoria.