

# Disseny de Sistemes Operatius

Enginyeria en Informàtica - FIB - UPC

## PRÁCTICA 1: Programación sobre sistemas operativos y microkernels

---

### 1. Objetivos de la práctica

El objetivo de esta primera práctica es familiarizarse con la programación de aplicaciones que trabajen directamente sobre un sistema operativo o un microkernel. Se trata de utilizar las abstracciones más básicas de Solaris, Linux y Mach, las librerías de threads y las llamadas a sistema respectivas, comprender qué objetos representan y cómo funcionan, y realizar las pruebas necesarias para ser capaces de desarrollar aplicaciones y servidores sobre estos sistemas.

Para ello hemos preparado una serie de ejemplos de aplicaciones sobre Mach que deben ser compilados y probados, estudiar las llamadas al kernel que realizan y los resultados o errores que provocan. Además de estas pruebas, la práctica consiste en modificar estos ejemplos para realizar las mismas tareas (o equivalentes) sobre Solaris y Linux y responder a las preguntas de este enunciado.

Tendremos por tanto tres entornos de trabajo. Uno con la versión microkernel de OSF/1: Mach 3.0, sobre procesadores Pentium de Intel. El segundo, el sistema operativo Solaris sobre la arquitectura SPARC de SUN Microsystems. Y el tercero, Linux sobre arquitectura Pentium. Es una componente más de la práctica el llegar a tener claras las diferencias entre los distintos interfaces utilizados y comparar el funcionamiento de las aplicaciones en cada entorno.

### 2. Entornos de trabajo en Solaris y Linux

Los entornos de Solaris y Linux ofrecen distintas versiones del interfaz de UNIX. Actualmente ambos sistemas son compatibles con los interfaces de System V y BSD. No obstante, cada sistema suele introducir algunas llamadas distintas de gestión más básica de sus abstracciones.

Así por ejemplo, en Linux disponemos de la llamada *clone* para crear flujos (realmente se crea un proceso con recursos compartidos) y en Solaris disponemos de una serie de llamadas para gestionar los *lwp* (*lightweight processes*), que son los flujos de nivel sistema.

Podéis trabajar con Solaris en la máquina Solfoc y con Linux en cada PC local que utilicéis para trabajar en el laboratorio.

En ambos sistemas encontraréis algunos comandos que os pueden ayudar a conocer más a fondo las abstracciones básicas que ofrecen. Tanto en Linux como en Solaris,

\$ ps

\$ top

incorporan nuevas opciones para obtener más información sobre los procesadores, procesos o flujos.

Además, Solaris proporciona diversos comandos nuevos como,

\$ psrinfo,

\$ pbind

y también nuevas llamadas al sistema como *processor\_bind* o *processor\_info*.

### 3. Entorno de trabajo en Mach 3.0

La versión microkernel de Mach sobre la que trabajaremos es Mach 3.0. Para ello disponemos de una máquina conectada a la red del Departamento de Arquitectura de Computadores: **fossas.ac.upc.es**. Tiene cuatro procesadores Intel Pentium Pro.

En esta máquina tenéis los mismos usernames que en Solfoc.

Existen unos comandos específicos de Mach 3.0 que permiten consultar las características de los objetos de Mach: **ms**, **pinfo**, **vminfo**. También disponéis en esta máquina de las páginas de manual de las llamadas a sistema de Mach.

### 4. Instalación y ejecución de los ejemplos

Los ejemplos para Mach 3.0 se hallan en el directorio **/users/soft/edso/P1** de **fossas.ac.upc.es**, en formato tar comprimido con gzip. También los podéis encontrar en Solfoc, en:

```
/home/soft/assig/dso/P1
```

Para cada tema existe un fichero, que se puede desmontar con:

```
$ gunzip -c /users/soft/edso/P1/t1-task-thread.mach3.0.tar.gz | tar xvf -
```

que creará el directorio **t1-task-thread** y situará el Makefile y los fuentes dentro de él.

Debido a que se hacen con más frecuencia backups de los discos de Solfoc que de los de **fossas.ac.upc.es**, se recomienda guardar copia de las prácticas en Solfoc. Recordad también que los discos de los PC's donde realizáis las prácticas se borran al reinicializar el sistema.

### 5. Enunciado de las modificaciones a realizar

Hemos organizado los ejemplos en temas, que corresponden con algunos de los capítulos del libro "Programming under Mach", by J. Boykin, D. Kirschen, A. Langerman, S. LoVerso, Addison-Wesley Publishing Company, Inc., 1993. Los ejemplos no son exactamente los del libro; han sido modificados para que puedan correr en las versiones actuales de Mach.

Escoged preguntas relativas a **como mínimo cuatro temas** y que sumen, por grupo de prácticas, **más de quince puntos** de entre las propuestas siguientes:

#### Tema 1: task y thread, threads de kernel

T1.1 - Creación de procesos y tasks en Linux, Solaris y Mach 3.0. En Mach 3.0, probad la creación de threads en otra task. Comparad las diferencias que encontréis entre los procesos UNIX y las tasks de Mach 3.0. (2 puntos)

T1.2 - En Mach 3.0, en el programa *mmat* haced que el thread principal se bloquee dentro del kernel con *suspend* y el último que acabe le despierte con un *resume*. Portad el ejemplo a Solaris, utilizando *lwps* (*lightweight processes*). (4 puntos)

T1.3 - En Mach 3.0, haced que los threads calculadores de *mmat* no terminen, sino que se queden en un *suspend* y el primer thread visualice por pantalla su estado (por ejemplo, con `'ps -m'`). Portad el ejemplo a Solaris utilizando *lwps* (*lightweight processes*). (3 puntos)

T1.4 - Cuál es el límite de número de threads de kernel por task en las máquinas de que disponemos? Comprobadlo experimentalmente. (1 punto)

#### Tema 2: port y message, comunicación entre procesos

T2.1 - Comprobad, en Mach 3.0, que se pueden unir las dos llamadas *send* y *receive* del cliente en una sola llamada al kernel. (2 puntos)

T2.2 - Comprobad el funcionamiento del paso de derechos SEND\_ONCE de un puerto en Mach 3.0. Modificad el servidor para que intente devolver dos respuestas a una petición del cliente, cuando éste sólo ha dado derechos de SEND\_ONCE. (3 puntos)

T2.3 - Qué ocurre cuando el tamaño del mensaje es distinto del esperado? Por ejemplo, que el servidor retorne un mensaje mayor que el buffer preparado para recibirlo. Verificad qué se recibe en cada caso. (4 puntos)

T2.4 - Qué capacidad de buffering en el kernel tienen los puertos de Mach. Es posible cambiar el tamaño del buffer o el número de mensajes encolados? Las llamadas de *ipc* bloquean al thread? (3 puntos)

### Tema 3: región, memoria virtual

T3.1 - Añadid la función que lista las regiones de memoria y sus atributos (`show_vm_regions` del fichero `vm_regions.c`) al programa *inherit\_demo*. Mostrad las regiones y sus atributos cada vez que se modifiquen. Buscad alguna llamada de Solaris y Linux que permita obtener una información equivalente en estos sistemas. (2 puntos)

T3.2 - Comprobad si al reservar memoria virtual con las mismas características, el kernel unifica las regiones contiguas. Utilizad la rutina *show\_vm\_regions* para mostrarlo. (2 puntos)

T3.3 - Si se pide una región de memoria virtual, se protege contra escritura y se intenta escribir en ella, qué ocurre si no se ha programado un puerto de excepciones? Portad este ejemplo a Solaris i Linux. (3 puntos)

T3.4 - Cread una task, asignadle memoria virtual y copiad parte de vuestro espacio de direcciones en la nueva task. Listad las características de las regiones que se crean en la nueva task con *show\_vm\_regions*. Comprobad que no se puede acceder fuera del espacio asignado. Comprobad si en Solaris o Linux podemos asignar memoria a un proceso distinto. (3 puntos)

### Tema 4: MIG, Mach Interface Generator

T4.1 - Modificad las definiciones del interfaz descrito en formato MIG para que con una sola rutina remota se pueda pedir la hora tanto en texto como numéricamente. Modificad adecuadamente el cliente y el servidor. (2 puntos)

T4.2 - Añadid un parámetro a una de las rutinas de la interficie del servidor, de forma que se pasen derechos sobre un puerto (enviar, enviar una vez o recibir). Haced que el cliente le pase los derechos en el mensaje y comprobad en el servidor que se reciben correctamente (llamada `port_type`). (4 puntos)

T4.3 - Estudiad cómo los *stubs* generados por MIG comprueban todo tipo de posibles errores en los mensajes. Se crea cada vez un puerto de respuesta? Escribid un interfaz equivalente utilizando RPCgen en Solaris/Linux y comparad los *stubs* generados. (3 puntos)

T4.4 - Habitualmente, junto a las rutinas generadas por MIG, se suministra una rutina de librería llamada `mach_msg_server`, que realiza la función de recepción del mensaje, llama a la función decodificadora de MIG y devuelve la respuesta. Modificad el servidor para utilice este mecanismo. (2 puntos)

### Tema 5: Pthreads, Cthreads, threads de usuario

T5.1 - Evaluad el overhead de la creación de threads, modificando la multiplicación de matrices de forma que cada thread realice más o menos operaciones y, por tanto, se creen, más o menos threads. Realizad la prueba en los tres sistemas. (2 puntos)

T5.2 - Jugando con el valor de la variable de entorno `PARALLEL` y el tamaño de los ficheros a buscar, verificad si puede ocurrir que haya threads que se encuentren con que no hay nada de trabajo para ellos. Realizad la prueba en los tres sistemas. (2 puntos)

T5.3 - Modificad la llamada `pthread_create` para que reciba unos atributos construidos por vosotros. En ellos, poned prioridades diferentes a los threads. Realizad la prueba en los tres sistemas. (2 puntos)

T5.4 - Verificad cuántos threads de kernel se crean cuando una aplicación crea bastantes threads de usuario (Pthreads y Cthreads). Se corresponden los valores? Porqué? Realizad la

prueba en los tres sistemas. (2 puntos)

## Tema 6: Excepciones

T6.1 - Haced que el gestor de la excepción (pexc1.c) no termine al tratar la excepción sino que quede en un bucle tratando excepciones. Comprobad que si no se soluciona, la excepción se repite infinitamente. (2 puntos)

T6.2 - La rutina de atención a la excepción, soluciona el problema que la ha provocado de alguna manera? Comprobad que la ejecución continúa, aunque el resultado puede no ser correcto. Podemos realizar el tratamiento de una excepción de dirección ilegal en Solaris/Linux? Cómo? (3 puntos)

T6.3 - Modificad los programas para que la excepción sea la de dirección ilegal. En el tratamiento de la excepción se le puede dar la memoria virtual que le falta e intentar la continuación del proceso. (4 puntos)

T6.4 - Modificad el programa pexc2.c para ver qué pasa si la excepción se soluciona durante el tratamiento correspondiente al thread. Verificad si el tratamiento de la task también se activa. (4 puntos)

## 6. Entrega de las prácticas

Recordad que debéis entregar las respuestas a las preguntas relativas a como mínimo cuatro temas y que sumen, por grupo de prácticas, más de quince puntos. Las prácticas se entregarán enviando un mail con un único fichero, que contenga en primer lugar el 00-README que se halla en fossas:/users/soft/edso/doc, los fuentes desarrollados, los fuentes del juego de pruebas, el informe y los resultados del juego de pruebas. Este fichero estará ensamblado con el comando shar que se halla en /home/soft/assig/dso/bin. Podéis utilizar:

```
$ alias shar '/home/soft/assig/dso/bin/shar'    en Solfoc
$ /usr/local/bin/shar                          en fossas ya está en el PATH.
```

Para que la lectura de los documentos entregados sea más fácil y comprensible, os encarecemos a que los ficheros entregados sigan una estructura regular, organizada en directorios y nombres de ficheros que indiquen lo que contienen. Así, al desmontar el shar, nos deberíamos encontrar con el siguiente árbol:

```
dsoNN/00-README
dsoNN/Informe
dsoNN/Mach3.0/TemaX/README
dsoNN/Mach3.0/TemaX/Makefile
dsoNN/Mach3.0/TemaX/P1.1.h
dsoNN/Mach3.0/TemaX/P1.1.c
dsoNN/Mach3.0/TemaX/....
dsoNN/Mach3.0/TemaY/....
dsoNN/Solaris/TemaZ/README
dsoNN/Solaris/TemaZ/Makefile
dsoNN/Linux/TemaZ/P5.2...
```

y se debería poder compilar todos los fuentes de un tema dado, simplemente entrando en su directorio y ejecutando make. Como veis, hay un informe global de toda la práctica, con comentarios sobre los resultados de todos los temas, y un README en el directorio del tema con información de qué ficheros se pueden encontrar en ese directorio y cómo se lanzan los juegos de pruebas (comando, parámetros, etc).