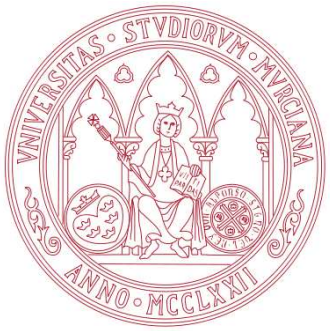

Universidad de Murcia

Facultad de Informática

TÍTULO DE GRADO EN
INGENIERÍA INFORMÁTICA

Fundamentos de Computadores

Tema 4: Introducción a los sistemas operativos

Boletín de autoevaluación de teoría / problemas

CURSO 2020 / 21

Departamento de Ingeniería y Tecnología de Computadores

Área de Arquitectura y Tecnología de Computadores

Índice general

Ejercicios resueltos

1. Señala cuál de las siguientes afirmaciones sobre el sistema operativo Linux es correcta:

   a) Diseñado inicialmente como alternativa a Windows en los procesadores Intel, actualmente corre exclusivamente sobre éstos.

   b) Su código fuente es abierto.

   c) Utiliza únicamente su propio sistema de ficheros, por lo que el manejo de ficheros de otros sistemas operativos se hace complicado.

2. A pesar de sus diferencias, las distintas distribuciones de Linux comparten siempre:

   a) Un mismo sistema de archivos (salvo quizá diferencia de versión).

   b) Un mismo entorno gráfico (salvo quizá diferencia de versión).

   c) Un mismo núcleo del SO (salvo quizá diferencia de versión).

3. ¿Cuál de las siguientes sería la definición más correcta de proceso?

   a) El código que ejecuta el procesador en un periodo de tiempo durante el cual el usuario interactúa con el sistema.

   b) Un código que se ejecuta cuando estamos en modo núcleo.

   c) Un programa en ejecución.

4. Los principales ficheros ejecutables de aplicaciones de usuario de un sistema Linux se encuentran en los directorios ____ y ____, mientras que los comandos más típicos de superusuario están en el directorio ____. Por su parte, el directorio del que cuelgan los respectivos directorios de trabajo de todos los usuarios del sistema es ____. Las librerías del sistema (código ejecutable compartido por distintos programas) se encuentran en su mayoría bajo los directorios ____ y ____. Finalmente los ficheros de configuración global de la mayoría de programas se encuentran bajo el directorio ____, y el directorio ____ se usa como directorio temporal (especificar en todos los casos las rutas absolutas).

5. En Linux, todos los procesos son descendientes (directos o no) del proceso llamado ____, que es el primer proceso iniciado por el ____ al arrancar el sistema. Los procesos pueden terminar de modo voluntario, ejecutando la llamada al sistema ____, o "matados" desde el exterior, usando la llamada ____.

6. De las siguientes afirmaciones, señala cual es falsa:

   a) Es el planificador de Linux el que se encarga de gobernar el cambio de tarea.

   b) Linux permite la multitarea, siempre y cuando existan varios procesadores o núcleos en la CPU.

   c) Los procesos bloqueados en una E/S provocan la cesión de la CPU a otro proceso.

7. A la parte software del sistema operativo que controla un dispositivo de E/S se le denomina ____ del dispositivo, mientras que al chip que controla físicamente dicho dispositivo se le denomina ____ del dispositivo.

8. Una de las mayores ventajas del uso del intérprete de comandos queda evidenciada porque:

   a) Se pueden combinar comandos para hacer cosas muy potentes.

   b) Es la única manera de comprimir ficheros para que ocupen menor tamaño en disco.

   c) No hay otra manera de cambiar los permisos a un directorio o fichero.

9. El *prompt* del sistema en Linux es:

   a) Una indicación de usuario, máquina y directorio actual que aparece en el `bash`, y que invita a introducir nuevos comandos.

   b) Un estándar de tratamiento de interrupciones software.

   c) El puntero que movemos con el ratón cuando cargamos el entorno gráfico.

10. El sistema operativo de un PC:

    a) Es una parte del hardware (almacenada en ROM) que toma el control al arrancar el ordenador.

    b) Es una capa software que toma el control al arrancar el ordenador, y administra el entorno de ejecución de programas.

    c) Es un programa que sirve únicamente para inicializar el ordenador y que, al cargar el entorno gráfico, termina su ejecución.

11. Terminología usada en sistemas operativos: A un programa en ejecución se le denomina habitualmente ____, mientras que al periodo de tiempo durante el cual un usuario interactúa con el sistema se le llama ____. Aquellos sistemas operativos que permiten su uso simultáneo por varios usuarios se denominan sistemas operativos ____. Para interactuar con un sistema operativo, se puede hacer a través de un GUI, siglas que significan exactamente ____ (en castellano ____), o bien tecleando órdenes en un terminal, método denominado interfaz de ____.

12. ¿Cuál de las siguientes afirmaciones sobre X-Window es correcta?

    a) Permite ejecutar aplicaciones gráficas sólo en el computador local.

    b) Permite ejecutar aplicaciones gráficas en un computador remoto, siempre que en aquel se esté ejecutando un servidor X-Window.

    c) Permite ejecutar aplicaciones gráficas en un computador remoto, siempre que en nuestra máquina local se esté ejecutando un servidor X-Window.

13. Explica brevemente la labor del planificador (*scheduler*) de un sistema operativo, y qué relación tiene con las interrupciones a la CPU por parte del reloj del sistema.

14. Explicar así mismo qué hace el *scheduler* en el momento en que a) llega una petición de E/S por parte de un proceso, y b) cuando el dispositivo de E/S termina la transacción.

15. Un *driver* de dispositivo:

    a) Ejecuta código que se comunica con la controladora del dispositivo.

    b) Es un chip que controla físicamente el dispositivo.

    c) Es siempre un proceso de usuario independiente del *kernel*.

16. POSIX es:

    a) Un estándar para definir interfaces de llamadas al sistema.

    b) Un estándar de tratamiento de interrupciones software.

    c) Un estándar de tratamiento de interrupciones hardware.

17. De las siguientes afirmaciones, señala cual es falsa:

    a) Cada proceso del sistema tiene su propia memoria virtual, que el SO puede mapear a distintas direcciones de memoria física.

    b) Todos los procesos del sistema administran directamente su propia memoria física.

c) Parte del espacio de memoria virtual de un proceso puede estar en disco.

18. El comando linux para comprobar el tamaño ocupado por el subárbol completo que cuelga de un directorio es ____, mientras que el comando que se emplea para saber el espacio libre de todos los sistemas de ficheros montados en el sistema es ____. El comando para cambiar los permisos de un archivo es ____, mientras que ____ y ____ son, respectivamente, los comandos empleados para cambiar el usuario y el grupo al que pertenece un archivo o directorio. Naturalmente, estos dos últimos comandos no pueden ser ejecutados por cualquier usuario, sino que necesitan ser ejecutados con los permisos de un usuario especial, llamado ____.

19. Dado un directorio en Linux con permisos '`r-x------`', señalar la única cierta de las siguientes afirmaciones:

    a) Nadie puede modificar el contenido de un archivo existente contenido en dicho directorio.

    b) Nadie puede crear archivos nuevos dentro de dicho directorio.

    c) Sólo el propietario puede crear archivos nuevos dentro de dicho directorio.

20. El carácter que se emplea en la línea de comandos (entre el comando y el archivo involucrados) para redireccionar la salida estándar de un comando a un archivo es ____, mientras que si lo que se desea es que el comando tome su entrada estándar de un archivo (en lugar de desde el teclado) entonces se usa el carácter ____. Si al redireccionar la salida a un archivo queremos que éste, en caso de existir, no se sobreescriba, sino que el texto redireccionado se añada al final del que ya había, entonces emplearemos la secuencia de caracteres ____. Para redireccionar la salida de error de un comando a un archivo, borrando sus contenidos anteriores, emplearemos la secuencia ____, y para redireccionar esa misma salida de error, pero en este caso añadiendo texto al final del archivo en lugar de sobreescribirlo, si este existía, emplearemos la secuencia ____.

21. En Linux, los planificadores de procesos y memoria, y los drivers de E/S se ejecutan:

    a) Como una cierta parte (subrutina) del proceso `ps`.

    b) En el modo usuario de la CPU.

    c) En el modo núcleo de la CPU.

22. El comando `programa > f1.txt 2>> f2.txt` en Linux:

    a) Copia el fichero programa en los ficheros destino `f1.txt` y `f2.txt`.

    b) Ejecuta el comando `programa`, redirigiendo su salida normal tanto al fichero `f1.txt` (al que sobreescribe), como al `f2.txt` (en el que escribe al final).

    c) Ejecuta el comando `programa`, redirigiendo su salida normal al fichero `f1.txt` (al que sobreescribe), y su salida de mensajes de error al `f2.txt` (en el que escribe al final).

23. **[*]** Sólo uno de los siguientes comandos de Linux se puede considerar un filtro de caracteres. ¿Cuál?

    a) El comando `who`.

    b) El comando `tr`.

    c) El comando `ps`.

24. **[*]** Ejemplos de filtros de líneas serían el comando ____, que sirve para quedarnos en la salida sólo con aquellas líneas de la entrada que contienen un determinado patrón, el comando ____, que sirve para ordenar (alfabética o numéricamente) las líneas de entrada, o los comandos ____ y ____, que sirven, respectivamente, para quedarnos en la salida con las primeras o últimas n líneas de la entrada. El filtro ____, sin embargo, es un filtro adecuado si lo que queremos es quedarnos en la salida sólo con un determinado rango de columnas del texto de entrada. Por último, el filtro ____ puede considerarse de caracteres, porque es el empleado cuando queremos realizar un tratamiento de algún tipo (borrado, sustitución, etc.) carácter a carácter.

25. A las llamadas al sistema por parte de un proceso también se las conoce como:

    a) Interrupciones software.

    b) Interrupciones hardware.

    c) Excepciones.

26. Indicar la orden completa que emplearías para comprimir todos los ficheros de tu directorio de trabajo y de sus subdirectorios en un fichero llamado `trabajo.tgz`.

27. Utiliza la orden `tar` para crear un fichero llamado `bin.tgz` con los contenidos que cuelgan del directorio `/bin`. A continuación, utiliza el comando `split` (consultar en el manual su funcionamiento) para dividir dicho fichero en fragmentos con un tamaño no superior a 700 KB cada uno (por ejemplo, para que cupiesen sin problema en disquetes de 720 KB cada uno). Finalmente, usar el comando `cat` con la salida redireccionada al fichero `bin2.tgz` para reconstruir el fichero original. (Sí, ahora suena un poco a chino, pero en su día, cuando no existían las memorias USB de muchos gigas, teníamos que hacer esto para poder sacar copias de seguridad de ficheros grandes ☺).

28. Establece, mediante la(s) órden(es) adecuada(s) de la línea de comandos, los permisos adecuados para que el fichero `~/temp/solucion.txt` sólo pueda ser borrado por el propietario, leído y modificado por el propietario y el grupo, y leído por el resto de usuarios.

29. Un directorio que pertenece al usuario `juan` y al grupo `users`, tiene los permisos `rwxr-x---`. Dentro de ese directorio existen únicamente dos ficheros, `fich1` y `fich2`, también pertenecientes al usuario `juan` y al grupo `users`, y con permisos `rw-rw-r--` y `r-xr-xr-x`, respectivamente. Supongamos también que existe otro usuario `pedro` que también pertenece al grupo `users`. Determina si las siguientes afirmaciones son verdaderas o falsas:

    - El usuario `juan` puede modificar `fich1`.
    - El usuario `pedro` puede borrar `fich1`.
    - Cualquier usuario puede ejecutar `fich2`.
    - El usuario `juan` puede cambiar los permisos de `fich2`.
    - El superusuario puede modificar `fich1`.

30. **[*]** Genera en un fichero llamado `salida.txt` el contenido del directorio `/usr/lib` mostrando, para cada fichero, su nombre y su número de nodo-i (un número identificador único para cada fichero del sistema), mediante la órden adecuada de la línea de comandos. **Nota**: comprueba qué hace la opción `-i` del comando `ls`.

31. **[*]** Busca todos los ficheros del propietario `root`, a partir del directorio `/usr/bin`, que se hayan modificado hace más de una semana y que tengan un tamaño inferior a 5 KB, mediante la(s) órden(es) adecuada(s) de la línea de comandos.

32. **[*]** Observa la profundidad máxima de tu subárbol personal (a partir de tu directorio *home*). **Nota**: usa `find` con la opción `-printf "%d \n"` (consultar manual)..

Ejercicios propuestos

1. Define brevemente, indicando las similitudes y diferencias entre ellas, los conceptos de interrupción, excepción y *trap*.

2. Explica brevemente, apoyándote con un ejemplo, el mecanismo que usa el sistema operativo para gestionar una llamada al sistema que involucre una entrada/salida con un dispositivo. ¿Tiene esto algo que ver con la multitarea? ¿Y con las interrupciones? Razona brevemente tus respuestas.

3. Explica brevemente qué es el redireccionamiento al ejecutar un proceso.

4. Comenta brevemente todos los trucos (comandos, pulsaciones de teclas, etc.) de la línea de comandos de Linux que sepas para moverte y reutilizar el historial de órdenes, acelerar el tecleo de comandos, etc.

5. Experimentar, con un terminal de comandos delante, con todas las posibilidades de los comodines para referirse a conjuntos de ficheros (asterisco, interrogación, llaves, corchetes, etc.). Para ello, crear un subdirectorio nuevo, meterse en él, y generar una buena cantidad de ficheros con nombres distintos (p.e., usando el comando `touch nombre_fichero`), pero que compartan ciertas características comunes en su nombre, para ir probando mediante comandos `ls [expresion con comodines]` las distintas posibilidades.

6. Crear un subárbol con una jerarquía de ficheros relativamente amplia (pongamos, por ejemplo, hasta cuatro niveles de profundidad, con hasta dos o tres subdirectorios por cada directorio, y un mínimo de un archivo por cada subdirectorio). Usar para ello los comandos `mkdir` (para ir creando los subdirectorios), `cd` (para ir moviéndonos posteriormente a ellos, ganando profundidad), `touch` (al estilo del ejercicio anterior, para ir creando nuevos archivos con el nombre deseado) y `cp` (para ir copiando en la jerarquía creada archivos ya existentes desde otros subdirectorios del sistema, por ejemplo `/bin` o `/usr/bin`). Después, moviéndonos por la jerarquía creada con `cd`, ir abriendo con el editor de textos `gedit` varios de los ficheros vacíos generados anteriormente con `touch`, y llenarlos con caracteres aleatorios para obtener ficheros de diferentes tamaños.

   Experimentar entonces con las opciones `-size` y `-printf` del comando `find` sobre dicha jerarquía, realizando diferentes búsquedas de ficheros con un tamaño menor o mayor que un valor especificado, imprimiendo para cada fichero encontrado no sólo su ruta y nombre, sino también el tamaño y la profundidad (consultar el manual de `find` y usar la búsqueda con el carácter `/` en la misma para buscar palabras de interés).

   Ejemplo: `find dir_test -size -80c -printf "El fichero %p tiene %s bytes (<80), pertenece al usuario %u, y está a una profundidad de %d en el subárbol de directorios.\n"`

7. Supongamos que la salida de un comando `ls -l` ejecutado en un determinado directorio es la siguiente:

```
drwxr-x--- 2 juan becarios   4096 jun  15 20:02 masdocs
-rw-r--r-- 1 juan becarios 219451 jun  15 20:01 Manual.pdf
drwxrwxrwx 1 juan becarios     50 jun  15 20:03 otrosdocs
-rw-r--r-- 2 juan todos    801945 jun  15 19:57 video.avi
```

   a) Detalle lo más posible toda la información disponible sobre la entrada de la primera línea (`masdocs`).

   b) Escribe el comando que `juan` tendría que utilizar para cambiar los permisos del archivo `Manual.pdf` para que pudiese ser escrito y leído por el usuario, sólo leído por los miembros de su grupo, y ni leído ni escrito por el resto de usuarios.

   c) Expresar con palabras lo que haría este comando:
   `find /otrosdocs -type f -size +20k -iname "[A-F]*.?"`

8. Indica los comandos de Linux que han de utilizarse para:

   a) Saber en qué directorio nos encontramos y quién es el usuario actualmente conectado en el terminal.

   b) Ir desde cualquier directorio a nuestra cuenta en `/home`.

   c) Movernos al directorio `/home/juan` desde el directorio `/home/pedro`, usando una ruta relativa y una ruta absoluta.

   d) Listar los contenidos del directorio `/bin`, incluyendo permisos, dueños, tamaño, etc., y almacenarlos en un fichero; suponiendo que estamos en `/home/alumno` y utilizando tanto una ruta absoluta como una relativa.

   e) Encontrar todos los ficheros regulares (e.d., no directorios) que cuelgan del subdirectorio `lib` (ubicado dentro del subdirectorio `usr`, éste último colgando directamente de la raíz), y cuyo nombre acaba con una `p` o una `P`.

   f) Explica la diferencia, y qué ocurriría, entre lanzar las líneas de comandos "`gedit & kate &`" y "`gedit ; kate`". Dado el primer caso: si introducimos el comando `fg` ¿qué ocurriría? ¿y si seguidamente pulsásemos `Ctrl-z`?, finalmente, ¿cómo matarías ambos procesos?

9. Supongamos que la salida de un comando `ls -l` ejecutado en un determinado directorio es la siguiente:

```
-rw-r--r-- 1 juan grupo_estudio 22151 sep 16 12:40 examen_feb_11.odt
drwxr-x--- 2 juan grupo_estudio  4096 oct  2 20:01 Examenes_pasado
```

   a) Indica toda la información que puedas sobre los contenidos de dicho directorio, a la vista de todos los metadatos mostrados por el comando.

   b) Describe qué comandos habría que utilizar para que `antonio`, que pertenece al grupo `grupo_estudio` pudiera mover el fichero `examen_feb_11.odt` al directorio `Examenes_pasado`.

   c) Describe qué comando(s) habría que utilizar para que el fichero `examen_feb_11.odt`, ya en el directorio `Exámenes_pasado`, pudiera ser modificado por cualquier miembro de su grupo.

10. Indicar los comandos de Linux que han de utilizarse para:

   a) Saber qué usuario somos.

   b) Saber cuál es nuestro directorio de trabajo.

   c) Movernos al directorio de inicio.

   d) Crear el directorio `/home/antonio` desde el directorio `/home/pedro`, usando una ruta relativa.

11. Supongamos que la salida de un comando `ls -l` ejecutado en el directorio `xpp` es la siguiente:

```
-rw-r--r-- 1 usuario usuario 22151  sep 28 12:40 ppito
d--------- 2 usuario usuario  4096  oct  7 20:01 sinpermisos
```

   a) Indica toda la información que puedas sobre los contenidos de dicho directorio, a la vista de todos los metadatos mostrados por el comando.

   b) Escribe el comando que habría que utilizar para cambiar los permisos del archivo `ppito` para que pudiese ser escrito y leído por el propietario, sólo leído por los miembros de su grupo, y ni leído ni escrito por el resto de usuarios.

   c) Indica qué habría que hacer para que el usuario `usuario` y el usuario `juan`, perteneciente también al grupo `usuario`, pudieran borrar el fichero `ppito`.

12. Un directorio llamado `examenes` que pertenece al usuario `juan` y al grupo `profesores`, tiene los permisos '`rwx rwx r-x`'. Dentro de ese directorio existen únicamente dos ficheros, llamados `examen1.txt` y `examen2.txt`, también pertenecientes al usuario `juan` y al grupo `profesores`, y con permisos '`rw- rw- r-`' y '`r-x r-x r-x` respectivamente. Supongamos también que existen otros usuarios llamados `pedro` y `jaime` que también pertenecen al grupo `profesores`. Determina si las siguientes afirmaciones son verdaderas o falsas, justificando brevemente cada respuesta:

    a) El usuario `juan` puede modificar `examen2.txt`.

    b) Los usuarios `pedro` y `jaime` pueden borrar `examen2.txt`.

    c) Los usuarios `pedro` y `jaime` pueden modificar `examen2.txt`.

    d) Cualquier usuario puede leer `examen1.txt`.

    e) Los usuarios `pedro` y `jaime` pueden crear y borrar nuevos ficheros en el directorio `examenes`.

    f) Cualquier usuario puede ver el contenido del mencionado directorio.

    g) Cualquier usuario convertido en superusuario podría borrar el directorio `examenes`.

    h) Cualquier usuario puede borrar o crear ficheros nuevos en el directorio `examenes`.

13. Indica toda la información que puedas sobre la siguiente salida de un comando `ps -Af` ejecutado desde el terminal, que contiene las líneas siguientes:

```
UID    PID     PPID   C STIME TTY   TIME     CMD
[...]
alumno 26158    1     2 19:56 pts/0 00:00:01 gnome-terminal
alumno 26160   26158  2 19:56 pts/0 00:00:03 bash
alumno 6180    26160  0 19:56 pts/0 00:00:00 ps -Af
```

Soluciones a los ejercicios resueltos

1. *Solución:* b)
2. *Solución:* c)
3. *Solución:* c)
4. *Solución:* `/bin`, `/usr/bin`, `/sbin`, `/home`, `/lib`, `/usr/lib`, `/etc`, `/tmp`.
5. *Solución:* `systemd`, núcleo (también llamado *kernel*), `exit`, `kill`.
6. *Solución:* b)
7. *Solución:* Driver, controladora.
8. *Solución:* a)
9. *Solución:* a)
10. *Solución:* b)
11. *Solución:* Proceso, sesión, multiusuario, Graphical User Interface, Interfaz Gráfico de Usuario, línea de comandos.
12. *Solución:* c)
13. El *scheduler* permite el uso compartido de la(s) CPU(s) entre distintos procesos. Para ello, periódicamente (un elevado número de veces por segundo), el reloj del sistema interrumpe al proceso actual, y entra a ejecutarse código del núcleo (*kernel*), que cambia el contexto para pasar a ejecutar otro proceso. Así, todos los procesos van avanzando con sensación de simultaneidad, a pesar de que en realidad se están ejecutando por turnos (aunque, en todo caso, también puede haber paralelismo real si el procesador dispone de varios núcleos).
14. Los procesos que realizan una llamada al sistema para realizar una operación de E/S (p.e., lectura de disco) quedan temporalmente bloqueados, y el *scheduler* cede entonces automáticamente la CPU a otro(s) proceso(s). Mientras dicha transacción no ha terminado, el planificador mantiene en ejecución en la CPU a otros procesos que no están bloqueados en una E/S (por turnos, como se aclaraba en la solución anterior), siempre intentando optimizar el uso de este recurso. Finalmente, cuando el dispositivo de E/S ha terminado su tarea, el proceso que realizó la llamada al sistema es desbloqueado, y pasa de nuevo a "competir" con el resto de procesos no bloqueados por la CPU.
15. *Solución:* a)
16. *Solución:* a)
17. *Solución:* b)
18. `du`, `df`, `chmod`, `chown`, `chgrp`, superusuario (también llamado `root`).
19. *Solución:* b)
20. *Solución:* >, <, >>, 2>, 2>>.
21. *Solución:* c)

22. *Solución:* c)
23. *Solución:* b)
24. *Solución:* `grep`, `sort`, `head`, `tail`, `cut`, `tr`
25. *Solución:* a)
26. *Solución:* `tar czf trabajo.tgz ~/*`
27. *Solución:*
```
tar czvf bin.tgz /bin/
split -d -b 1400k bin.tgz part
cat part* >bin2.tgz
```
28. *Solución:*
```
chmod 755 ~/temp
chmod 664 ~/temp/solucion.txt
```
29. *Solución:*
    a) Sí.
    b) No (`juan` puede borrarlo, `pedro` sólo puede modificarlo).
    c) No, porque aunque el permiso de ejecución está activo en ese fichero para todos los usuarios del sistema, dichos usuarios no tienen acceso en lectura para el directorio en el que está contenido (sin embargo, los usuarios `juan` y `pedro`, así como el resto de posibles usuarios del grupo `others`, sí que podrían).
    d) Sí.
    e) Sí.
30. *Solución:* `ls -i /usr/lib > salida.txt`
31. *Solución:* `find /usr/bin -type f -user root -mtime +7 -size -5k`
32. *Solución:* `find ~ -type d -printf "%d \n"`