1º de Grado en Ingeniería Informática (Grupos 1, 2, 3 y 4)

Examen de Fundamentos de Computadores (convocatoria de enero)

29 de enero de 2015

SEGUNDO PARCIAL (TEMAS 4, 5 y 6)

Apellidos, Nombre:_____ **DNI**:_____ **Grupo(1/2/3/4)**:__

Parte II: cuestiones teórico-prácticas *(35.0%; puntuación indicada en cada apartado)*

C1. *(1.00 puntos)* Indicar brevemente las similitudes y diferencias que hay entre estos tres posibles comandos de Linux (**Nota.-** No se trata de una secuencia de comandos, se trata de comandos distintos para realizar una tarea deseada de formas alternativas):

a) `cp video.avi video2.avi`

b) `ln video.avi video2.avi`

c) `ln -s video.avi video2.avi`.

Indicar también, para cada uno de los tres casos por separado, lo que ocurrirá exactamente si tras ejecutar el comando correspondiente, se ejecuta el comando `rm video.avi`. Suponer que el tamaño del archivo `video.avi` es relativamente grande, p.e. alrededor de un gigabyte, y especificar, para la ejecución particular de cada uno de los tres comandos, qué ocurrirá con el espacio usado en el disco.

C2. *(1.5 puntos)* La pila en el x86-64:

a) ¿Qué es la pila? Defínela muy brevemente. Explica su estructura en cuanto al uso que se hace de ella. ¿Qué registros del procesador deben usarse para manejar la pila en el x86-64?

b) Describe al menos tres de los usos más típicos de la pila en un programa.

c) Indicar al menos tres ejemplos de instrucciones que hagan uso (explícito o implícito) de la pila, indicando las respectivas acciones que, en concreto, realizan dichas instrucciones.

C3. *(1.00 puntos)* Completa los huecos que aparecen en la siguiente tabla de distintas subredes (se puede contestar, si se quiere, en la misma tabla):

IP	***Máscara***	***Subred***	***Broadcast***	**Dir *Host* 1º**	**Dir *Host* último**
155.54.1.130	255.255.255.128	155.54.1.128	155.54.1.255	155.54.1.129	155.54.1.254
216.58.210.15	255.255.255.0	216.58.210.0	216.58.210.255		
192.168.4.61	255.255.255.224				192.168.4.62
155.54.60.99	255.255.0.0				

Parte III: ejercicios boletines *(35.0%; puntuación indicada en cada apartado)*

P1. *(1.00 puntos)* Rellenar el siguiente texto (indicando en el examen la correspondiente referencia al hueco para cada respuesta):

"Una vez arrancado el ordenador y lanzado el terminal de comandos (`bash`) aparece un *prompt* como el siguiente: `Carlos@linux_casa:~$`. La virgulilla ('~') me indica que estoy en el directorio __(a1)__, pero si quiero asegurarme de cuál es mi directorio actual usaré el comando __(a2)__. Si deseo moverme al directorio `apuntes` que cuelga del directorio actual usaré el comando __(a3)__, y para saber los ficheros y directorios que están en éste, así como sus permisos, tamaño, fecha de modificación etc., usaré el comando __(a4)__, aunque si lo que me interesa es saber la fecha de creación usaré el comando __(a5)__. De ese mismo directorio cuelga otro directorio que se llama `Fundamentos` y de él cuelgan otros 6 directorios llamados `tema_1`, `tema_2`, etc. Si quiero encontrar, sin moverme de donde estoy, todos los ficheros en formato `pdf` que se llamen `tema1-boletin-practicas.pdf`, `tema2-boletin-practicas.pdf`, etc. hasta `tema4-boletin-practicas.pdf`, sin saber exactamente en qué subdirectorio están (pero bajo `apuntes/Fundamentos`) usaré el comando __(a6)__. Una vez localizados los anteriores ficheros deseo copiarlos en un directorio que aún he de crear que se llamará `examen` (colgará del directorio `apuntes`) y para crearlo utilizo el comando __(a7)__). Si deseo copiar con un sólo comando todo el directorio `Fundamentos` y todo lo que cuelgue de él al recién creado directorio `examen`, entonces utilizaré el comando __(a8)__. Finalmente deseo tener en un fichero de texto llamado `todo.txt` todos los nombres de ficheros (sólo ficheros) que cuelgan desde `Fundamentos`, para ello hago uso del comando con redireccionamiento:__(a9)__."

P2. *(1.5 puntos)* Considérese la siguiente sesión `gdb` (los puntos suspensivos `[...]` indican que se ha suprimido la parte de la salida que no nos interesa para el ejercicio):

```
(gdb) l
[...]
2       int array[5] = {100,200,300,400,500};
3       int main() {
4           int i;
5           for(i=4;i>=0;i--)
6               array[i] = funcion(i);
[...]
(gdb) disassemble main
[...]
0x400585 <+8>:        movl   $0x5,-0x4(%rbp)
0x40058c <+15>:       jmp    0x4005ae <main+49>
0x40058e <+17>:       mov    -0x4(%rbp),%eax
[...]
0x400598 <+27>:       callq  0x4005f0 <funcion>
[...]
0x4005aa <+45>:       subl   $0x1,-0x4(%rbp)
0x4005ae <+49>:       cmpl   $0x0,-0x4(%rbp)
0x4005b2 <+53>:       jns    0x40058e <main+17>
[...]
```

```
(gdb) x/48bx 0x400585
0x400585 <main+8>:    0xc7  0x45  0xfc  0x05  0x00  0x00  0x00  0xeb
[...]
0x4005ad <main+48>:   0x01  0x83  0x7d  0xfc  0x00  0x79  0xda  0xc7
(gdb) x/20bx array
0x601050 <array>:     0xYY  0x00  0x00  0x00  0xZZ  0x00  0x00  0x00
0x601058 <array+8>:   0x2c  0x01  0x00  0x00  0x90  0x01  0x00  0x00
0x601060 <array+16>:  0xf4  0x01  0x00  0x00
```

En base a dicha sesión, rellenar todos los huecos del texto que va a continuación (indicando en el examen la correspondiente referencia al hueco para cada respuesta):

> "El código depurado en la sesión manipula una tabla de __(a1)__ elementos de tipo entero de 32 bits, que ocupa exactamente __(a2)__ bytes en memoria, y que comienza en la dirección exacta __(a3)__. La función `funcion`, por su parte, comienza exactamente en la dirección __(a4)__. Justo al inicio de la ejecución del programa, el valor de byte que aparece sustituido en negrita con **`0xYY`** será, en realidad, el valor __(a5)__ (suponer que el convenio de almacenamiento utilizado es *little endian*), mientras que el valor de **`0xZZ`** será en realidad __(a6)__. El código desensamblado que aparece en la figura se corresponde con un código de alto nivel que __(a7)__ (dar aquí una explicación concisa pero completa de lo que hace dicho código). En particular, la instrucción `movl $0x5,-0x4(%rbp)` dentro de ese código (y sabiendo que en algún lugar previo del programa se puso `%rbp` apuntando al marco de pila de la función `main`), exactamente realiza lo siguiente: __(a8)__, mientras que la instrucción `cmpl $0x0,-0x4(%rbp)` se encarga de: __(a9)__. La instrucción ubicada en la dirección `0x40058c` es de tipo __(a10)__, mientras que la ubicada en la dirección `0x4005b2` es de tipo __(a11)__. Podemos también afirmar que la instrucción `cmpl`, una vez ubicada en memoria, ocupa exactamente __(a12)__ bytes, cuyos valores concretos son __(a13)__ (expresar aquí los bytes correspondientes tal y como aparecen en el listado), que están comprendidos entre la dirección __(a14)__ y la __(a15)__, ambas inclusive."

P3. *(1.00 puntos)* Responda a las siguientes preguntas sobre redes de ordenadores:

- **a)** ¿Cuál es la dirección de red y de *broadcast* de un equipo con la IP y máscara siguientes: 155.54.49.1/20?
- **b)** Dada la subred del equipo anterior, especifique la primera IP válida para asignársela a un router, e indique **justificadamente** si éste intervendría o no para poder alcanzar desde este equipo a otro con IP 155.54.50.0/20.
- **c)** ¿Cuántas interfaces de red válidas, en total, se podrían conectar como máximo a esa subred?
- **d)** Rellene los huecos de cada apartado (d1)-(d4), considerando que ambos comandos Linux se han ejecutado en ese mismo equipo:

```
user@host:~$ ifconfig
eth0  Link encap: Ethernet     direcciónHW        00:1A:BC:2E:00:00
      Direc. inet:__(d1)__     Difus.: __(d2)__    Másc: 255.255.240.0
```

```
user@host:~$ route
Destino    Pasarela  Genmask          Indic  Métric   Ref   Uso   Interfaz
__(d3)__   *         255.255.240.0    U      0        0     0     eth0
Default    __(d4)__  0.0.0.0          UG     0        0     0     eth0
```

- **e)** Ídem para los apartados (e1), (e2) y (e3): *"Si dispongo de una red `155.54.0.0/16` tendré un total (incluyendo posibilidad de router) de hasta __(e1)__ dispositivos a conectar (es decir, interfaces de red como máximo); si configurara entonces todos los equipos de la misma con la máscara `255.255.240.0`, obtendría __(e2)__ subredes de __(e3)__ interfaces de red (incluyendo los respectivos routers) cada una".*

SOLUCIÓN C1:

a) El primer comando genera una copia en el disco duro del archivo `video.avi`, llamada `video2.avi`. Dicha copia ocupará exactamente el mismo tamaño que el vídeo original (la entrada al directorio no ocuparía más, excepto si se rebasan los 4KB del directorio donde se hace la copia (*)), duplicándose por tanto el espacio necesario (otro GB adicional). Al tratarse de dos archivos independientes, el borrado de `video.avi` con el correspondiente comando `rm` no tendrá ningún efecto sobre el archivo copiado (se borrará y liberará el espacio ocupado por `video.avi`, mientras que `video2.avi` seguirá perfectamente accesible).

b) El segundo comando, sin embargo, genera una nueva entrada de directorio llamada también `video2.avi`, igual que el comando anterior, y cuyo contenido será exactamente el mismo (es decir, podremos acceder a él exactamente igual que en el caso 1). Pero en este caso el espacio usado en el disco será mucho menor (en realidad casi siempre nada más, pues es espacio que se modifica de los 4KB del directorio, a menos que se sobrepasen esos 4KB), puesto que el contenido del vídeo en sí no estará duplicado. Se trata del denominado *enlace de tipo duro*. Si entonces se borra el archivo original `video.avi` con el comando `rm` no hay ningún problema, puesto que el archivo `video2.avi` seguirá conteniendo los datos del vídeo original.

c) El tercer caso es similar al segundo, en cuanto a que se genera también una entrada nueva de directorio llamada `video2.avi`, con los mismos contenidos, pero sin duplicar el espacio de disco necesario. Sin embargo, en este último caso, al tratarse de un *enlace de tipo simbólico*, el efecto del comando de borrado `rm` sobre el archivo original `video.avi` será que el contenido del vídeo se borrará irremediablemente del disco duro, quedando el enlace simbólico `video2.avi` "colgando" sin apuntar a nada.

(*) Aunque ésto último en realidad depende de los detalles de implementación de cada sistema de archivos concreto que se pueda estar utilizando (ext3, ext4, FAT, NTFS, etc.).

SOLUCIÓN C2:

a) Es una zona de la memoria principal que se maneja de una manera especial. La pila es una estructura de datos en la que existe un puntero para conocer la "cima de la pila", o posición de memoria donde se almacenará el siguiente dato a apilar. La pila crece (aumenta su contenido) desde direcciones altas a direcciones bajas de la memoria principal (a esta operación se le llama "apilar" y decrece (se sacan datos de ella o "desapilar") hacia direcciones altas. Se dice que es una estructura LIFO. Los punteros o registros donde se almacenan los valores donde está la cima de la pila son el `%rsp`, y el `%ebp` para apuntar a los marcos de pila de las distintas funciones, que contienen sus respectivos parámetros, variables locales, direcciones de retorno, etc..

b) 1.– Para guardar la dirección de retorno en una llamada a una subrutina (procedimiento o función). Se deja en la pila el valor del puntero de instrucciones o contador de programa mediante una instrucción CALL y se recupera con la instrucción RET.

2.- Para pasarle valores a los procedimientos (especialmente cuando nos quedamos sin registros del procesador para este propósito), lo que se hace desde el procedimiento llamador.
3.- Para salvaguardar los registro que usaremos dentro del procedimiento y que no deseemos "machacar". Se hace con instrucciones PUSH al principio del procedimiento y devolvemos esos valores a los registros con instrucciones POP.
4.- Para reservar un espacio temporal en memoria para variables locales al procedimiento (se puede hacer al crear el marco de pila).
5.- Para pasarle los valores de los resultados calculados en el procedimiento al procedimiento principal (desde el procedimiento llamado).
6.- También se usa la pila en Subrutinas de Atención a las Interrupciones (pero no se vio en clase)

c) *Implícitamente:* CALL, RET, PUSH, POP, LEAVE (también INT e IRET, PUSHF y POPF, no visto en clase).
Explícitamente: `mov` con el registro `%rbp` (haciendo apuntar `%rbp` a la pila).

CALL *subrutina*: Apila el `%rip` y salta a la subrutina.
RET: Desapila la en el `%rip` la cima de la pila.
PUSH *registro*: apila el *registro* en la cima de la pila, decrementando convenientemente el valor de `%rsp`.
POP *registro*: Justo lo contrario, saca el valor de la cima de la pila y lo guarda en el *registro* incrementando el puntero de la pila.
LEAVE: Abandona el marco de pila despilando los registro previamente apilados.

SOLUCIÓN C3:

IP	***Máscara***	***Subred***	***Broadcast***	**Dir *Host* 1º**	**Dir *Host* último**
155.54.1.130	255.255.255.128	155.54.1.128	155.54.1.255	155.54.1.129	155.54.1.254
216.58.210.15	255.255.255.0	216.58.210.0	216.58.210.255	**216.58.210.1**	**216.58.210.254**
192.168.4.61	255.255.255.224	**192.168.4.32**	**192.168.4.63**	**192.168.4.33**	192.168.4.62
155.54.60.99	255.255.0.0	**155.54.0.0**	**155.54.255.255**	**155.54.0.1**	**155.54.255.254**

SOLUCIÓN P1:

a1) */home/Carlos*
a2) `pwd`
a3) `cd apuntes`
a4) `ls -l`
a5) `ls -lU`

a6) `find Fundamentos -name "tema[1-4]-boleti*.pdf" (p. ejem.)`
a7) `mkdir examen`
a8) `cp -R Fundamentos examen`
a9) `find /home/Carlos/apuntes/Fundamentos -type f > todo.txt` o
`find ./Fundamentos -type f > todo.txt`

SOLUCIÓN P2:

a1) 5.
a2) 20.
a3) `0x601050`.
a4) `0x4005f0`.
a5) `0x64` (100 en decimal).
a6) `0xc8` (200 en decimal).
a7) Inicia la variable `i` del bucle `for` a 5, en el cuerpo del bucle llama a la función `funcion`, y al final del bucle decrementa la variable `i`, comprueba si es menor que cero, y en caso contrario cierra el bucle.
a8) Inicializa la variable `i` (que está en la pila) con el valor 5.
a9) Compara el valor de la variable `i` (que está en la pila) con el valor 0.
a10) Salto incondicional.
a11) Salto condicional.
a12) 4 bytes.
a13) `0x83 0x7d 0xfc 0x00`.
a14) `0x4005ae`.
a15) `0x4005b1`.

SOLUCIÓN P3:

a) *Red*:155.54.48.0 *Broadcast*: 155.54.63.255
b) Primera IP válida de router: 155.54.48.1. En el trayecto indicado no interviene el router, puesto que la máquina destino está en la misma subred que la máquina origen.
c) 2^{12}-2 = 4094.
d) d1) 155.54.49.1
d2) 155.54.63.255
d3) 155.54.48.0
d4) 155.54.48.1 (aunque en realidad valdría cualquiera entre 155.54.48.1 y 155.54.63.254, quitando por supuesto la del propio host, 155.54.49.1)
e) e1) 65534.
e2) 16.
e3) 4094.