Apellidos: _____ **Nombre:**_____ **DNI:**_____ **Grupo:** _____

1º de Grado en Ingeniería Informática – SÓLO 2º PARCIAL

Final de enero de Fundamentos de Computadores (temas 4, 5 y 6)

8 de enero 2017

Instrucciones para realizar el examen (tipo A)

- El tiempo disponible es de 2 horas.
- No olvide poner los apellidos y el nombre tanto en la hoja de examen como en los folios entregados.
- Para las preguntas tipo test, selecciónese una única respuesta en cada cuestión en el lugar habilitado para ello (señalando con una X en la tabla colocada al comienzo del test). El resto de preguntas se contestarán en folios.
- Cada dos respuestas incorrectas en el test anulan una correcta. Una pregunta sin contestar ni suma ni resta.
- Entregad tanto el enunciado del examen como los folios utilizados o no al acabar el examen.

Parte I: tipo test *(27%; 0.15 puntos por respuesta)*

A	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18
a																		
b																		
c																		

T1. Las funciones de un sistema operativo son:
- **a)** Chequear que memoria, CPU y dispositivos funcionen bien al arrancar el ordenador.
- **b)** Administrar CPU, memoria y dispositivos.
- **c)** Administrar la correcta transferencia de datos entre el puente norte y puente sur, y entre los distintos dispositivos conectados a ellos.

T2. En cuanto a la potencia o características de la máquina o a su propósito principal de cómputo, los sistemas operativos se pueden clasificar en:
- **a)** Linux, Windows o Mac OS.
- **b)** Para propósito general, servidores, tiempo real, integrados, supercomputadoras...
- **c)** De código abierto o de pago.

T3. En Linux, si quisiera buscar los archivos ejecutables de los comandos más habituales, p.e., `ls`, `ps`, `rm`, `cp`, etc., los encontraría en el directorio:
- **a)** /bin/.
- **b)** /usr/bin/.
- **c)** /etc/.

T4. La diferencia entre un enlace físico (*duro*) y uno simbólico (*blando*) radica en que:
- **a)** El enlace duro "gastará" tanto espacio de más en disco duro cuanto sea el tamaño de los datos del fichero original enlazado (no así el enlace blando).
- **b)** El enlace simbólico no permite modificar el contenido del fichero sino solamente modificando el fichero enlazado original (no así el enlace duro).
- **c)** Podemos borrar cualquier enlace duro que, mientras que subsista uno, los datos serán accesibles, mientras que, en los enlaces blandos, si borramos el fichero enlazado original, los datos no serán accesibles.

T5. Una expresión que englobara a los nombres de un fichero que empezara por a, con tantos caracteres como fueren con la única condición que acabara en z antes de un punto, al que le siguieran dos o más caracteres, sería:
- **a)** `a?z.**`
- **b)** `a*z.??`
- **c)** `a*z.?*?`

T6. Si queremos que se nos muestren todos los archivos (no directorios) del directorio `/tmp` y sus descendientes cuyo nombre sea exactamente "`examen.pdf`", tendremos que ejecutar la orden Linux:
- **a)** `ls -lR /tmp -type f -name "examen.pdf"`
- **b)** `find /tmp -type f -name "examen.pdf"`
- **c)** `find -r /tmp -type f -iname "examen.pdf"`

T7. ¿Cuál de estas frases acerca de las librerías estáticas y dinámicas en Linux es correcta?
- **a)** Un binario (ejecutable) que ha sido enlazado dinámicamente responde a la orden `ldd` informando de las librerías enlazadas.
- **b)** Un binario (ejecutable) que ha sido enlazado estáticamente responde a la orden `ldd` informando de las librerías enlazadas.
- **c)** Si no se indica lo contrario, el compilado y enlazado por defecto de un archivo fuente en lenguaje C, con la orden `gcc`, es de forma estática.

T8. La orden Linux que muestra todos los procesos en ejecución actualizándose en tiempo real es:
- **a)** `top`
- **b)** `ps -Af`
- **c)** `fg`

(sigue atrás)

T9. La instrucción `movq $0,(%rax)` del ensamblador del x86-64 sirve para:

a) Almacenar el valor cero en las 8 direcciones consecutivas de memoria apuntadas por la dirección contenida en el registro `%rax`.

b) Almacenar el valor cero en los 4 bytes consecutivos de la dirección de memoria contenida en el registro `%rax`.

c) Copiar el contenido del registro `%rax` de memoria al registro `$0` del procesador.

T10. Para apilar el contenido del registro `%eax` usaríamos el código ensamblador del x86-64 siguiente:

a) pushl %eax.

b) movl %eax, (%esp) y después subl $4, %esp.

c) Las dos formas serían válidas.

T11. Un compilador:

a) Traduce un código fuente escrito en un lenguaje de alto nivel a código ejecutable final.

b) Compila un módulo objeto con librerías estáticas o dinámicas para conseguir el ejecutable final.

c) Es un módulo del sistema operativo que es llamado por el cargador y que ayuda a éste a preparar la pila.

T12. La instrucción `ret` del repertorio del x86-64:

a) Es funcionalmente idéntica a una instrucción `jmp`.

b) Salta a la dirección contenida en la cima de la pila.

c) Salta a la dirección contenida en el registro `%rip`.

T13. En lo referente al ensamblador del x86-64:

a) Una etiqueta únicamente puede ser sinónimo de una dirección de memoria del segmento de instrucciones.

b) Una etiqueta únicamente puede ser sinónimo de una dirección de memoria del segmento de datos.

c) Una etiqueta es sinónimo de una dirección de memoria, bien del segmento de instrucciones o del de datos.

T14. Las siglas DHCP se refieren a:

a) La conversión de direcciones para permitir el uso de IPs privadas en subredes manteniendo la conectividad con Internet.

b) La determinación automática de los parámetros de la red en un host mediante un protocolo de configuración.

c) El servicio de traducción a/de nombres de host de/a direcciones IP.

T15. En redes, el termino NAT:

a) Significa Network Access Transport, y pertenece a la capa de transporte de la pila de protocolos.

b) Significa Network Address Translation y permite conectar varias interfaces a una única dirección IP pública.

c) Significa National Associated Telematics y es un organismo que vela a nivel local por el buen uso de internet.

T16. Considerando varios procesos de red tipo cliente en una misma máquina, que se han conectado a un proceso único en un servidor. Es cierto que:

a) Cada proceso cliente emplea un puerto distinto para cada comunicación.

b) El cliente ha de tener varias direcciones IP para que cada proceso envíe sus mensajes de origen con una IP distinta.

c) Es el servidor el que, al estar replicado, ofrece varias direcciones IP para que el cliente utilice direcciones IP distintas para cada proceso.

T17. Los protocolos SMTP, POP3 e IMAP:

a) Corresponden todos al nivel de aplicación.

b) El primero corresponde al nivel de red, el segundo al de transporte, y el tercero al de aplicación.

c) Los protocolos SMTP y POP3 sirven para enviar el correo, mientras que el protocolo IMAP se utiliza para recibir el correo.

T18. ¿Hasta cuántas subredes diferentes puedo llegar a obtener si, disponiendo de la red global 192.168.0.0/16, construyo subredes con máscara 255.255.240.0?

a) 16

b) 4096

c) Si la red global es 192.168.0.0/16, la máscara de red debe ser obligatoriamente 255.255.0.0

Apellidos: _____ **Nombre:**_____ **DNI:**_____ **Grupo:** _____

1º de Grado en Ingeniería Informática – SÓLO 2º PARCIAL

Parte II: cuestiones teórico-prácticas *(38%; puntuación indicada en cada apartado)*

C1. *(1.6 puntos)* Rellene los huecos de cada apartado sobre sistemas operativos:

"Al código ejecutable almacenado en disco para eventualmente ser ejecutado se le denomina **__($C_1$1)__**, *mientras que cuando éste pasa a ejecución se convierte en un* **__($C_1$2)__** *necesitado de recursos tales como* **__($C_1$3)__** , *memoria y dispositivos de E/S, que el SO se encarga de administrar. Existen dos visiones complementarias de un SO : a) como máquina* **__($C_1$4)__** *o virtual, encargado de ocultar la complejidad del* **__($C_1$5)__** *subyacente; b) como* **__($C_1$6)__** *de recursos, encargado de optimizar su uso. Una de las abstracciones básicas proporcionadas por el SO es la unidad lógica de almacenamiento para datos que necesitan persistencia, denominada* **__($C_1$7)__**. *Cada proceso tiene un estado que es cambiante: los valores de los registros del procesador (incluyendo el registro* **__($C_1$8)__** *que indica la dirección de la siguiente instrucción a ejecutar), de los datos en memoria, etc. El SO permite que los distintos procesos puedan compartir el uso de la(s) CPU(s) disponibles, lo cual se denomina* **__($C_1$9)__**. *De forma periódica, un reloj/temporizador genera una señal que hace que se detenga la ejecución del proceso actual, evento al cual llamamos* **__($C_1$10)__** *de tipo* **__($C_1$11)__**. *En este punto, pasa inmediatamente a ejecutarse el código del núcleo, que puede decidir ceder la CPU a otro proceso. Para ello, el SO guarda el estado del proceso actual y restaura el del siguiente proceso a ejecutar, a lo cual se le denomina* **__($C_1$12)__**. *Al código del núcleo encargado de decidir qué proceso se debe ejecutar en cada momento se le llama* **__($C_1$13)__**. *En otras ocasiones, es el propio proceso el que transfiere explícitamente el control al SO mediante una (no demasiado correctamente llamada) interrupción de tipo* **__($C_1$14)__**, *con el fin de solicitar un determinado servicio al núcleo. Por ejemplo, para primero abrir y luego leer un fichero en Linux, un programa de usuario debe realizar dos* **__($C_1$15)__** *(primero open* y luego *read*). *Finalmente, la ejecución de un proceso también puede verse detenida a causa de* **__($C_1$16)__**, *que son señales generadas internamente en la CPU ante una situación de error (p.ej., una división por cero).*

C2. *(1.0 puntos)* Modelo de programación del x86-64:

¿En qué grupos de instrucciones hemos clasificado el juego de instrucciones del x86-64? Para cada grupo pon un ejemplo de cada subgrupo de instrucciones que hemos estudiado, explicando exactamente qué hace tal instrucción.

C3. *(1.2 puntos)* Indicar los nombres de las cinco capas de la arquitectura de red usada por Internet, ordenadas de más alto a más bajo nivel. A continuación, indicar dos protocolos diferentes pertenecientes cada una de las capas primera, segunda y cuarta, más un sólo protocolo para la tercera capa (entendiendo de nuevo las capas ordenadas de más alto a más bajo nivel). Sólo será necesario expresar el significado concreto de las siglas de los dos protocolos mencionados en la segunda capa.

Parte III: ejercicios boletines *(35%; puntuación indicada en cada apartado)*

P1. *(1.0 puntos)* Suponiendo un *prompt* del sistema `alumno@lab:~$`, qué ordenes Linux habría que ejecutar en un sólo paso para poder realizar las siguientes acciones?

Nota: Siempre que sea necesario utilizar una ruta ésta deberá ser RELATIVA.

a) Mostrar por el terminal el contenido de un fichero oculto llamado `.preguntas.txt` ubicado en `/home/alumno/ejercicios`.

b) Renombrar como `ejemplo2.txt` un fichero que se llama `ejemplo.txt` y está ubicado en `/home/alumno/fc/ejercicios/ejemplo.txt`.

c) Listar el contenido de una carpeta `practicas` que cuelga de `/home/alumno/fc`.

d) Mover todo un directorio llamado `ANTIGUO` dentro de un directorio existente llamado `NUEVO`, ubicados ambos en `/home/alumno`.

e) Trasladar un archivo llamado `documento.pdf` ubicado en el directorio `/home/alumno/fc/practicas` al directorio presente.

f) Trasladar un fichero `hola.txt` ubicado en `/home/alumno/fc/ejemplos` al directorio `/home/alumno/fc/miscelanea`.

g) Duplicar un directorio `Practicas`, con todas sus subcarpetas y archivos, que cuelga de `/home/alumno` en `/home/alumno/fc` , con el nuevo nombre `Practicas_FC`.

h) Borrar totalmente el directorio `Practicas` del punto anterior.

i) Dado un achivo `foto1.jpg` dentro de una carpeta `/home/alumno/fotos`, dar únicamente permiso de lectura de ese archivo a `alumno` y a su grupo, y ningún permiso a nadie más.

j) Dar permisos a la carpeta anterior (`/home/alumno/fotos`) para que sólo `alumno` pueda crear y borrar archivos de ella, su grupo pueda acceder a los archivos del mismo, pero sin crear ni borrar archivos en ella, y el resto de usuarios no pueda acceder ni en lectura ni en escritura a dicha carpeta.

P2. *(1.5 puntos)* Considérese la siguiente sesión con el depurador `gdb` (aparecen subrayados los comandos tecleados por el usuario; el resto es la salida producida por el terminal), y rellenar todos los huecos del texto que va a continuación (indicando en el examen la correspondiente referencia al hueco para cada respuesta):

(Sigue atrás)

```
user@host:~/$ gdb ./main
(gdb) list
1   int array[4] = {-2,-6,-8,-14};
2   int i;
3   int main() {
4         for(i=0;i<4;i++)
5         array[i] = funcion(i);
6   }
7   int funcion(int parametro) {
8      return array[parametro] / 2;
9   }
(gdb) disassemble funcion
   0x0000000000400625 <+0>:     push   %rbp
   0x0000000000400626 <+1>:     mov    %rsp,%rbp
   0x0000000000400629 <+4>:     mov    %edi,-0x4(%rbp)
   0x000000000040062c <+7>:     mov    -0x4(%rbp),%eax
   0x000000000040062f <+10>:    cltq
   0x0000000000400631 <+12>:    mov    0x601070(,%rax,4),%eax
   0x0000000000400638 <+19>:    mov    %eax,%edx
   0x000000000040063a <+21>:    shr    $0x1f,%edx
   0x000000000040063d <+24>:    add    %edx,%eax
   0x000000000040063f <+26>:    sar    %eax
   0x0000000000400641 <+28>:    pop    %rbp
   0x0000000000400642 <+29>:    retq
(gdb) x/29bx 0x400625
0x400625 <funcion>:    0x55  0x48  0x89  0xe5  0x89  0x7d  0xfc  0x8b
0x40062d <funcion+8>:  0x45  0xfc  0x48  0x98  0x8b  0x04  0x85  0x70
0x400635 <funcion+16>: 0x10  0x60  0x00  0x89  0xc2  0xc1  0xea  0x1f
0x40063d <funcion+24>: 0x01  0xd0  0xd1  0xf8  0x5d
(gdb) x/16xb array
0x601070 <array>:      0xfe  0xff  0xff  0xff  0xfa  0xff  0xff  0xff
0x601078 <array+8>:    0xf8  0xff  0xff  0xff  0xf2  0xff  0xff  0xff
```

"El código depurado en la sesión anterior manipula una tabla de **__(P2 1)__** *elementos de tipo entero, que ocupará exactamente* **__(P2 2)__** *bytes en memoria, y que comienza en la dirección exacta* **__(P2 3)__** *. La función* `funcion`*, una vez ubicada en memoria, comienza exactamente en la dirección* **__(P2 4)__** *. Una vez ejecutado el código, el nuevo valor del byte que ocupa la dirección 0x601074 será* **__(P2 5)__** *, en lugar de su valor inicial, que era* **__(P2 6)__** *(expresar ambos bytes en formato 0xXX, con XX en hexadecimal). Por otro lado, la instrucción que se ejecutará justo antes del* `push rbp` *(que no aparece en el listado mostrado, pero que se puede fácilmente deducir) será* **__(P2 7)__** *, mientras que la instrucción tras cuya ejecución se volverá después para seguir ejecutando el programa por la instrucción siguiente a la anteriormente indicada será* **__(P2 8)__** *. Las instrucciones utilizadas para guardar en la pila el valor anterior del registro base de la pila, antes de modificarlo en la función, y posteriormente recuperarlo serán, respectivamente,* **__(P2 9)__** *y* **__(P2 10)__***. El registro que en todo momento contiene la dirección de la cima de la pila es el registro* **__(P2 11)__** *. Finalmente, el código máquina de la instrucción* `cltq` *ocupará exactamente* **__(P2 12)__** *bytes en memoria, cuyos valores exactos son* **__(P2 13)__** *, y se sitúan en el rango de direcciones de byte comprendido entre la dirección* **__(P2 14)__** *y la* **__(P2 15)__** *, ambas inclusive."*

P3. *(1.0 puntos)* Rellene los huecos de cada apartado sobre redes de ordenadores:

a) Dado un equipo con IP y máscara `155.54.202.190/22,` su dirección de red es **__(P3 a1)__** y su máscara de red es **__(P3 a2)__**.

b) En dicho equipo se han ejecutado dos comandos Linux. Rellene los huecos considerando que su router por defecto tiene asignada la primera IP (es decir, la dirección válida más baja) disponible de la subred:

```
user@host:~$ __(P3 b1)__ eth0
eth0  Link encap:Ethernet  direcciónHW 00:00:10:11:12:13
      Direc. inet:155.54.202.190  Difus.: __(P3 b2)__ [...]



user@host:~$ __(P3 b3)__
Destino    Pasarela    Genmask         Indic  Métric  Ref  Uso  Interfaz
_(P3 b4)_  0.0.0.0     255.255.252.0   U      100     0    0    eth0
default    _(P3 b5)_   0.0.0.0         UG     100     0    0    eth0
```

c) Rellene los huecos de cada apartado (c1), (c2) y (c3): *"Si dispongo de una red global* `155.54.1.0/24` *tendré un total (incluyendo posibilidad de router) de hasta* **_(P3 c1)_** *interfaces de red; si configurara entonces todos los equipos de la misma con la máscara* `255.255.255.224`*, obtendría* **_(P3 c2)_** *subredes de* **_(P3 c3)_** *interfaces de red (incluyendo los respectivos routers) cada una."*

Apellidos: _____ **Nombre:**_____ **DNI:**_____ **Grupo:** _____

Soluciones a SÓLO 2º PARCIAL:

Test:

Sólo 2º parcial A

A	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18
a			X				X	X	X	X	X					X	X	X
b	X	X				X						X		X	X			
c				X	X								X					

Cuestiones:

C1: *(1.6 puntos)*

$C_1$1: Programa
$C_1$2: Proceso
$C_1$3: CPU
$C_1$4: Extendida
$C_1$5: Hardware
$C_1$6: Administrador
$C_1$7: Fichero
$C_1$8: Contador de programa o puntero de instrucciones
$C_1$9: Multiprogramación o multitarea
$C_1$10: Interrupción o interrupción del *timer*
$C_1$11: Hardware
$C_1$12: Cambio de contexto
$C_1$13: Planificador o *scheduler*
$C_1$14: Software
$C_1$15: Llamadas al sistema o *syscall*
$C_1$16: Excepciones

C2: *(1.0 puntos)*

Transparencias de teoría tema 5, páginas 27-32 y 40:

1. Aritmético-lógicas: ejemplo `add %eax, %ebx`
2. Movimiento de datos: ejemplo `mov %eax, (%ebx)`
3. Salto incondicional: ejemplo `jmp label`
4. Salto condicional: ejemplo `jle label`
5. Soporte de procedimientos: ejemplo `call funcion`

C3: *(1.2 puntos)*

Transparencias de teoría tema 6, página 16:
1. Capa de Aplicación, protocolos HTTP y SMTP (por ejemplo).
2. Capa de Transporte, protocolos TCP (Transmission Control Protocol) y UDP (User Datagram Protocol).
3. Capa de Red, protocolo IP.
4. Capa de Enlace, protocolos IEEE 802.3 (Ethernet) y 802.11 (WiFi) (por ejemplo).
5. Capa Física.

Prácticas:

P1: *(1.0 puntos)*

Los comandos siguientes constituyen una posible solución a cada apartado, aunque en algunos casos puede haber alguna otra solución igualmente válida:

a) `cat ejercicios/.preguntas.txt`
b) `mv fc/ejercicios/ejemplo.txt fc/ejercicios/ejemplo2.txt`
c) `ls -l fc/practicas`
d) `mv ANTIGUO NUEVO`
e) `mv fc/practicas/documento.pdf .`
f) `mv fc/ejemplos/hola.txt fc/miscelanea`
g) `cp -r Practicas fc/Practicas_FC`
h) `rm -r Practicas`
i) `chmod 440 fotos/foto1.jpg`
j) `chmod 750 fotos/`

P2: *(1.5 puntos)*

$P_2$1) 4
$P_2$2) 16
$P_2$3) 0x601070
$P_2$4) 0x400625
$P_2$5) 0xfd
$P_2$6) 0xfa
$P_2$7) call funcion
$P_2$8) retq
$P_2$9) push %rbp
$P_2$10) pop %rbp
$P_2$11) %rsp
$P_2$12) 2
$P_2$13) 0x48 y 0x98
$P_2$14) 0x40062f
$P_2$15) 0x400630

P3: *(1.0 puntos)*

P_3a1) 155.54.200.0
P_3a2) 255.255.252.0

P_3b1) ifconfig
P_3b2) 154.54.203.255
P_3b3) route -n
P_3b4) 155.54.200.0
P_3b5) 155.54.200.1

P_3c1) 254
P_3c2) 8
P_3c3) 30