Apellidos: **Nombre:** **Matrícula:**

Examen Teórico
(1/3 de la nota final)

105000016 - Programación para Sistemas
Grado en Ingeniería Informática (2009)

Lenguajes y Sistemas Informáticos e Ingeniería de Software
Facultad de Informática
Universidad Politécnica de Madrid

Curso 2011/2012 - Julio 2012

Normas

- El examen teórico (**1/3 de la nota final**) puntúa sobre **12 puntos**.
- La duración es de **una hora**.
- Se deberá tener el DNI o el carnet de la UPM en lugar visible.
- No olvidar rellenar **apellidos, nombre y número de matrícula** en cada hoja.
- La solución se proporcionará antes de la revisión.
- La fecha prevista de publicación de calificaciones, a través del Aula Virtual (Moodle) de la asignatura, es el **17 de julio**.
- La fecha prevista de revisión del examen es el **18 de julio**, a las 16:00 en la sala 2319. Se confirmará a través del Aula Virtual (Moodle) de la asignatura.

Cuestionario

(1 punto) 1. Observar la siguiente sesión Bash:

```
$ cd /tmp
$ mkdir foo
$ ( cd foo; ); pwd
```

¿Cuál es la salida estándar del último mandato ejecutado?

A. `/tmp/foo`

B. `/tmp`

(1 punto) 2. La orden `rm -foobar` borrará correctamente el fichero con nombre `-foobar`.

A. Verdadero. **B. Falso.**

(1 punto) 3. En el manual de Bash, se puede leer la siguiente descripción sobre la `[[]]`:

```
[[ exp ]]
    Devuelve un estado de 0 ó 1 dependiendo de la evaluación de la
    expresión condicional exp.
```

Dado el siguiente mandato Bash:

```
a=1; b=1
[[ $a != $b ]]  &&  echo "a␣!=␣b"  ||  echo "a␣==␣b"
```

¿Cuál es su salida estándar?

A. a == b

B. `a != b`

(1 punto) 4. En el manual de Bash, se puede leer la siguiente descripción sobre la expansión de variables:

```
${parámetro:+palabra}
    Si parámetro está vacío o no está definido, no se sustituye nada;
de otro modo, se sustituye la expansión de palabra.
```

Escribir la salida estándar resultado de la ejecución de los siguientes mandatos Bash:

```
unset X
echo \"${X:+otro}\"
X=
echo \"${X:+otro}\"
X=uno
echo \"${X:+otro}\"
```

Solución:

```
""
""
"otro"
```

(1 punto) 5. Indique los tamaños en bytes de las siguientes variables declaradas en lenguaje C:

a) `int datos[] = { 1, 2, 3 };`

Nota: el tamaño de un carácter **char** es 1 byte, y el de un entero **int** es 4 bytes.

b) `char cadena[] = "Hola";`

Solución:

a) 12

b) 5

(1 punto) 6. Sea el siguiente fichero `prog.c` de código C:

```
#include<stdio.h>

int main() {
  char texto[41];

  fgets(texto, 40, stdin);
  printf("%s", texto);
  return 0;
}
```

Sea el fichero de 3 líneas `fich.txt` :

```
En un lugar
de la Mancha
de cuyo nombre
```

Indique la salida que produce la siguiente llamada al ejecutable `prog` (donde `prog` es el ejecutable asociado a `prog.c`):

```
cat fich.txt | ./prog
```

Solución: En un lugar

(1 punto) 7. Escriba un **typedef** con **struct** con nombre `complejo` adecuado para representar un número complejo.

Solución:

```
typedef struct
{
  double real;
  double imag;
} complejo;
```

(1 punto) 8. Indique la salida que produce la ejecución del siguiente código C:

```
#include <stdio.h>

int main() {
  int datos[] = { 1, 2, 3 };
  int *p = datos;

  *(p+1) = *p + *(p+2);
  printf("%d\n", *(p+1));
  return 0;
}
```

Solución: 4

(1 punto) 9. ¿Cuál de las siguientes funciones es apropiada para liberar memoria en C?

A. `delete`

B. `free`

C. `clear`

D. `remove`

(1 punto) 10. Se está realizando un programa `prog` que tiene `prog.c` como fichero fuente asociado. El ejecutable ha dado un error de ejecución y se quiere llamar al depurador `gdb` con un `core` (depuración *postmortem*).

Indique *todas* las acciones, especificando las llamadas concretas a compilador, sistema operativo, depurador, etc., que debe realizar para ello.

Solución:

- Llamada al compilador con flag -g:
  `gcc -g -Wall -ansi -pedantic prog.c -o prog`
- Llamada al comando bash ulimit para permitir la creación de ficheros core:
  `ulimit -c unlimited`
- Llamada al ejecutable prog para crear el fichero core (tras error de ejecución):
  `./prog`
- Llamada al depurador gdb con fichero core generado:
  `gdb prog core`

(1 punto) 11. Escriba el makefile que permita compilar una aplicación que consta de: (a) 3 archivos fuentes `procesar.c`, `leerjpeg.c` y `escribir.c`, y (b) un archivo cabecera `procesar.h` donde están las declaraciones de todas las funciones usadas por `procesar.c` y `escribir.c`. La función `main` está incluida en `procesar.c`. La aplicación usa una biblioteca del sistema denominada `libjpeg.a`. El nombre del ejecutable será `procesar`.

Solución:

```
CCFLAGS=-Wall -ansi -pedantic
procesar: procesar.o leerjpeg.o escribir.o
        gcc -o procesar procesar.o leerjpeg.o escribir.o -ljpeg
procesar.o: procesar.c procesar.h
        gcc $(CCFLAGS) -c procesar.c
leerjpeg.o: leerjpeg.c
        gcc $(CCFLAGS) -c leerjpeg.c
escribir.o: escribir.c procesar.h
        gcc $(CCFLAGS) -c escribir.c
```

(1 punto) 12. Sea el siguiente código en lenguaje C:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
char *p;

if ((p = (char *) malloc (1000000000000)) != NULL) {
  printf("A\n");
}
else {
  printf("B\n");
}
return 0;
}
```

Si al ejecutar dicho código aparece en la salida la letra `B`, comente brevemente en dos o tres líneas qué ha ocurrido en relación a la memoria dinámica.

Solución: El sistema operativo no ha podido proporcionar la memoria dinámica solicitada, siendo `NULL` el valor del puntero `p` tras la llamada a `malloc`.