

Examen

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 - Enero 2012

Normas

- El examen puntúa sobre **12 puntos**.
- La duración total del mismo es de **una hora y cuarto**.
- 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 al examen se proporcionará antes de la revisión.
- La fecha prevista de publicación de calificaciones es el **20 de enero**, y se realizará a través del Aula Virtual de la asignatura.
- La revisión del examen tendrá lugar el **23 de enero** a las 16:00 en la sala 2319.

Cuestionario

(1 punto) 1. Dado el siguiente mandato Bash (observar con cuidado):

```
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) 2. ¿Cuál de los siguientes comandos comprueba si el fichero /bin/ls es ejecutable?

A. [-r /bin/ls]

B. [-x /bin/ls]

C. [-e /bin/ls]

- (1 punto) 3. Escribir el valor que contiene la variable X tras la ejecución de los siguientes mandatos Bash:

```
read <<EOF
1 2 3
EOF
X=$REPLY
```

Solución:

```
1 2 3
```

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

```
${variable:-palabra}
    Si variable no está definida o su contenido es null, se
    realiza la expansión a palabra. En otro caso se realiza
    la expansión de variable.
```

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
otro
uno
```

(1 punto) 5. Escriba la salida que genera el siguiente programa en C:

```
#include <stdio.h>

int main (int argc, char*argv[])
{
    int vector[5]= {1,2,3,4,5};

    *(vector+2)=0;
    printf ("%d_ %d",vector[0],vector[2]);
    return 0;
}
```

Solución:

1 0

(1 punto) 6. Escriba la salida que genera el siguiente programa en C :

```
#include <stdio.h>

int main (int argc, char*argv[])
{
    int vector[5]= {1,2,3,4,5};

    if ( *(vector+4)==vector[3] ) {
        printf("IGUALES");
    } else {
        printf("DISTINTOS");
    }
    return 0;
}
```

Solución:

DISTINTOS

(1 punto) 7. Indique el comando del depurador gdb que permite monitorizar los cambios de una variable, parándose la ejecución del programa cuando se produzca una modificación de la variable.

Solución:

watch nombre_variable

- (1 punto) 8. Escriba el makefile que permita compilar una aplicación que consta de: (a) 2 archivos fuentes `procesar.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 escribir.o
    gcc -o procesar procesar.o escribir.o -ljpeg
procesar.o: procesar.c procesar.h
    gcc $(CCFLAGS) -c procesar.c
escribir.o: escribir.c procesar.h
    gcc $(CCFLAGS) -c escribir.c
```

(1 punto) 9. Escriba la salida que genera el siguiente programa en C:

```
#include <stdio.h>

int main (int argc, char*argv[])
{
    int *p,i;
    int x[5]={1,2,3,4,5};
    p=x;
    *p=9;
    p[1]=11;
    for (i=0;i<5;i++)
        printf(" %d_",x[i]);
}
```

Solución:

9 11 3 4 5

(1 punto) 10. Escriba una definición de tipos (**typedef**) y la correspondiente declaración de las variables para que las siguientes instrucciones tengan sentido:

```
a[5].frase[19] = '\0';
a[2].pdatos = (int *) malloc(100*sizeof(int));
```

Solución:

```
typedef struct
{
    char frase [20];
    int *pdatos;
} T_A;
T_A a[6];
```

(1 punto) 11. Escriba la salida que genera el siguiente programa en C:

```
#include <stdio.h>

int main (int argc, char*argv[])
{
    int *p;
    int x=7;
    int y=5;
    p=&x;
    *p=3;
    p=&y;
    *p=x;
    x=9;
    printf("%d_ %d_ %d", *p, x, y);
}
```

Solución:

3 9 3

(1 punto) 12. 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 para intentar localizar dónde en el código se produce el error.

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