



Apellido y Nombres	Legajo	# de Hojas

#### **Normas Generales**

Numere las hojas entregadas. **Lea detenidamente cada pregunta y consulte las dudas de interpretación que pudiesen surgir.** Complete en la primera hoja la cantidad total de hojas entregadas. Realice este parcial en lápiz, o con tinta color azul o negro. **No utilice rojo ni verde por favor.** Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con: Nombre, Apellido, Legajo. **Por favor entregar esta hoja y las restantes del tema junto al examen.**

#### **Sección teórica**

1. Explicar que es un filesystem, cual es su función, en que dispositivos de entrada salida se aplica, y mencionar algunos ejemplos conocidos.

2. Si tenemos el siguiente listado de un directorio:

```
-rw-r--r-- 1 user1 group1 110 2010-08-07 19:28 hola.txt
```

a) Se pide que escriba cómo quedarían los permisos si el usuario user1 ejecuta:

```
chmod 754 hola.txt
```

b) ¿Que ocurre si el comando anterior lo ejecuta el usuario root? ¿Y si lo ejecuta el usuario user4?

c) ¿Qué comando hay que ejecutar para ver el contenido del archivo "hola.txt" desde la consola?

d) ¿Con qué comando podemos eliminar el mismo archivo desde la consola?

3. ¿Que comandos permiten conocer la ubicación de un archivo en el filesystem? ¿Cuales son las principales diferencias entre ellos?

4. Explique que entiende por proceso. ¿es lo mismo proceso que programa?. ¿Que comandos conoce para visualizar los procesos desde el shell?

5. Si al ejecutar **gcc prueba.c -o prueba** recibimos la siguiente salida:

```
prueba.c:5: warning: return type of "main" is not "int"
prueba.c: In function "main":
prueba.c:10: warning: "return" with a value, in function returning void
/tmp/cc0GSL0s.o: In function `main':
prueba.c:(.text+0x1b): undefined reference to `log'
collect2: ld returned 1 exit status
```

a) ¿Se generó el ejecutable final?

b) ¿Qué información puede obtener a partir los mensajes recibidos?

c) ¿Qué solución propone para eliminar todos los inconvenientes?

d) Especificar en cada caso quién brinda el mensaje: precompilador, compilador o linker.

e) ¿Que opción le agregaría a la línea de compilación para poder utilizar un debugger (ddd, o kdbg) para depurar el programa?

6.

a) ¿Cuál es la diferencia entre un header y una librería?

b) ¿Qué contiene cada uno?

c) ¿Dónde especificamos la inclusión de un header en nuestro programa?

d) ¿Dónde y como especificamos la inclusión de una librería en nuestro programa?



Apellido y Nombres	Legajo	# de Hojas

7. Enumere las diferentes clases de librerías que se pueden generar. Explique las características de cada una, ventajas y desventajas, y con que herramienta de desarrollo se genera cada tipo de librería.

8. Si desde un programa invocamos la siguiente función:

```
int buscar(char *cadena, int cant, char* buffer);  
//busca la string cadena, de "cant" bytes de largo en buffer.
```

- a) ¿Por que área de memoria se pasan los tres argumentos a la función buscar?
- b) ¿Como implementa el compilador el pasaje de los argumentos?. Explicarlo lo mejor que pueda.
- c) La función "buscar" se implementa del siguiente modo:

```
int buscar (char *patron, int longitud, char* array)  
{  
    //aqui va el código  
}
```

Si estamos debugueando el programa con ddd o kdbg, y tenemos en la ventana de visualización la variable longitud, ¿cual será su valor luego de salir de la función y retornar al programa invocante?



Apellido y Nombres	Legajo	# de Hojas

### Sección Práctica

Escriba en sendos archivos fuente las siguientes funciones:

**unsigned int myStrLen (char \*str);**

Dada una *string*, pasada como argumento por referencia, calcula su longitud y la retorna en una variable del tipo *unsigned int*.


**int myStrEqu (char \*str, char \*orig);**

Recibe dos argumentos cada uno de los cuales es un puntero a *string* y devuelve en un tipo entero el valor 1 si ambas son iguales y 0 en caso contrario.

Con ayuda de estas funciones se desea desarrollar una solución que contribuya a un sistema automático de moderación de plataformas de comercio electrónico. **Este programa estará en otro archivo fuente.**

El programa recibe la “*string* a moderar” por línea de comandos. Ejemplo:

**\$ nombre\_programa “Homa mundo!! ¿Que tal?”**

 Atención a las comillas que encierran la *string*. Es mandatorio ingresarlo de esa forma!!! ¿Porque?

Ni bien se carga en memoria, el programa ejecutará la siguiente secuencia de operaciones:

1. Leerá la **lista** de palabras prohibidas por standard input. El ingreso finaliza cuando una cadena se compone solo de '\0', es decir cuando el usuario pulsó solo ENTER.
2. A continuación (es decir, una vez ingresados por stdin la lista completa de palabras prohibidas) el programa buscará en la *string* a moderar, cada una de las *strings* que componen la black list, y de encontrarla (puede repetirse varias veces la misma *string* prohibida), le reemplazará cada uno de sus caracteres por una “X”.
3. Una vez completada esta tarea, habrá obtenido la “*string* moderada”, la que imprimirá por standard output en líneas separadas junto con la cantidad de caracteres reemplazados por “X”.
4. Seguidamente esperará una tecla y de acuerdo a lo que haya pulsado el usuario:
  - Finalizará la ejecución, si se pulsa Q,
  - Presentará por standard output la black list ordenada alfabéticamente, si se pulsa S, retornando al mismo menú una vez completada la impresión.
  - Volverá pedir una nueva cadena de texto a moderar.

**Sugerencias:** Desarrollar una función **ModerateFromBlackList**, que reciba como parámetro la “*string* a moderar” y la referencia la *lista* de palabras prohibidas, y devuelva en un tipo entero la cantidad de caracteres a reemplazar. Esta función, apoyándose en las dos anteriores reemplazará cada palabra prohibida encontrada por una *string* que contenga una “X” por cada caracter reemplazado.

Usar gets () para manejar el ingreso de cadenas por teclado.

Incluir las bibliotecas del compilador que crea conveniente y las declaraciones de las funciones solicitadas.

### Condiciones de trabajo:

No se admitirá el uso de variables globales ni el uso de las funciones definidas en “string.h”.