

WUOLAH



PruebaAlien

www.wuolah.com/student/PruebaAlien



18119

exámenespracticas.pdf

exámenes_practicas



2º Sistemas Operativos



Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.



Exámenes, preguntas, apuntes.



Apellidos: [REDACTED]

Nombre: [REDACTED]

RECOMENDACIONES:

- Esta prueba sirve para evaluar los conocimientos adquiridos hasta el momento.
- No tengas miedo de preguntar algo que no comprendas; yo estoy aquí para ayudarlos.
- Ahora, olvida de los nervios y el estrés. ¡Sé que puedes hacerlo!

INSTRUCCIONES:

- Acceda a su cuenta personal especificando el código **examenubu16** y levantando **Ubuntu 16.04 de 32 bits**.
- No está permitido utilizar calculadora, móviles ni ningún dispositivo de comunicación o medio de comunicación.
- Únicamente se accederá a Prado para acceder al trabajo definido para esta prueba.
- Use únicamente las llamadas al sistema estudiadas en los guiones de prácticas del presente módulo.
- Los programas deben contener únicamente lo que se pide, se penalizará si incluyen funcionalidad no requerida.
- Programe la actuación adecuada ante las distintas situaciones de error que puedan ocurrir.

Ejercicio 1 [5 puntos]. Expliquemos en primer lugar la actuación de la orden `du`. Si estuviésemos en el shell `bash`, ejecutando la orden

```
du -k rutaCompletaArchivo
```

Se obtendría en la salida estándar el número de bloques (de 1024bytes) asignados a `rutaCompletaArchivo`

Se pide construir un programa en C llamado «bloques1.c» que recibirá cierto número de argumentos, cada uno de ellos es la ruta de un archivo. Para cada argumento recibido, hablemos del *i*-ésimo, deberá crearse un hijo que lance la ejecución la orden `du` pasándole como primer argumento «-k» y como segundo argumento `argv[i]`.


Tenemos entonces al proceso padre y a *argc* procesos hijos ejecutándose concurrentemente. Deberá programarse las redirecciones oportunas para que se comuniquen adecuadamente a través de un único cauce (elija usted con nombre o sin nombre) donde cada hijo deberá llevar su salida y donde el padre debe leer. El padre va leyendo cada dato y lo muestra en la salida estándar. Debe conseguirse la máxima concurrencia posible.

Ejercicio 2 [5 puntos]. Se pide construir un programa llamado «bloques2.c» que es una ampliación al anterior: Para cada dato que lea el padre del cauce (donde los hijos han ido escribiendo), el proceso padre guardará en un archivo en `/tmp` y con nombre «dato_N.txt» (donde *N* es un contador iniciado en 1 y que por cada lectura se va incrementando en una unidad), como contenido el valor de *argv[i]* ^{ruta.exe} sobre el que se ha calculado y el número de bloques asociado (No se desea imponer ningún orden, conforme el padre lea un dato se escribe en el archivo esta información).

Ejercicio extra [1 punto]. Construye un nuevo programa en C llamado «netflix.c». Deberá hacer lo mismo que `bloques2.c` (cópialo por completo), pero que reciba, además, como parámetro el nombre de una serie de Netflix. Cuando la `rutaCompletaArchivo` coincida con el nombre de archivo «netflix» cree (si no existe ya), un nuevo archivo regular en `/tmp` con el nombre «series_recomendadas.txt» y permisos `rwXr--r--`. Añada como contenido, siempre al final del archivo (use `\n` para separar) la nueva serie (parámetro `argv`).

Tip: `char*res=strrchr("/dir1/dir2/file", "/")` nos puede ayudar. Tendríamos `res="/file"`. Sin embargo, controla casos como `strrchr("otraruta", "/")` devuelve `NULL` porque «otraruta» no contiene «/». Si `res` no es `NULL` y `res="/file"`, si quieres puedes hacer `r=r+1` y tendrás `r="file"`, listo para compararlo con `argv[1]`.

2019-2020

 **UNIVERSIDAD DE GRANADA**
Departamento de Lenguajes y Sistemas Informáticos

Sistemas Operativos
Grado en Ingeniería Informática
Prácticas de Examen Módulo I

Ejercicio 1. Crea un programa en C llamado «ejercicio1.c» e implementa las siguientes tareas:

a) [1 punto] Realizar una exploración completa de todas las entradas del directorio «ficheros» proporcionado. Para ello, se puede utilizar la función «nftw». Filtrar por los siguientes criterios:

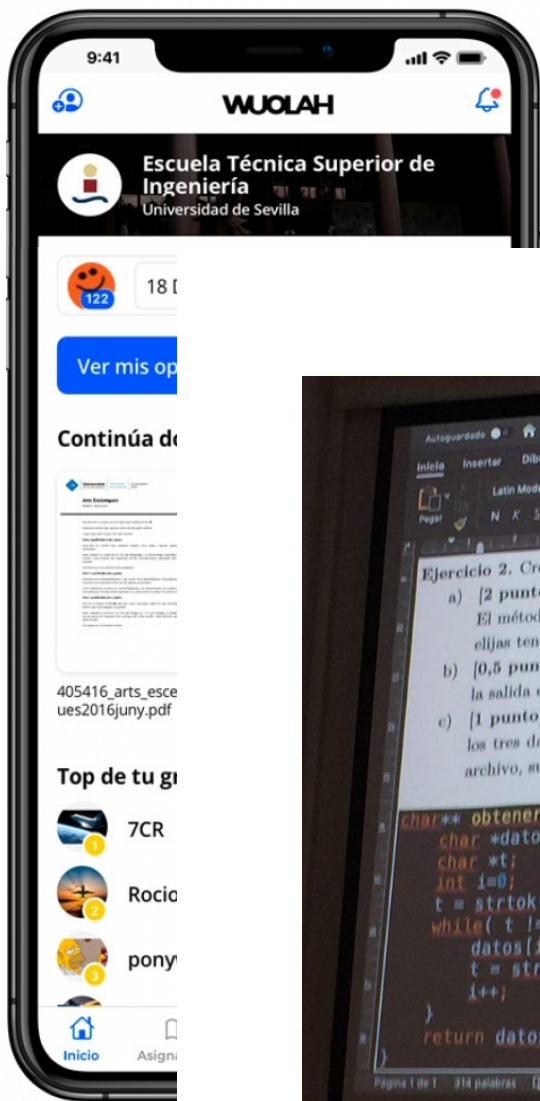
- Archivos regulares.
- Todos los permisos para usuario, ninguno para grupo y de escritura para otros.
- Longitud del nombre del archivo mayor que 7.

b) [1 punto] Por cada entrada seleccionada, escribir una cadena de caracteres por la salida estándar que siga este formato: nombre|permisos|tamaño. La salida resultante debería ser similar a esta:

```
permisos_702.txt|702|10399  
quijote.txt|777|1060259
```

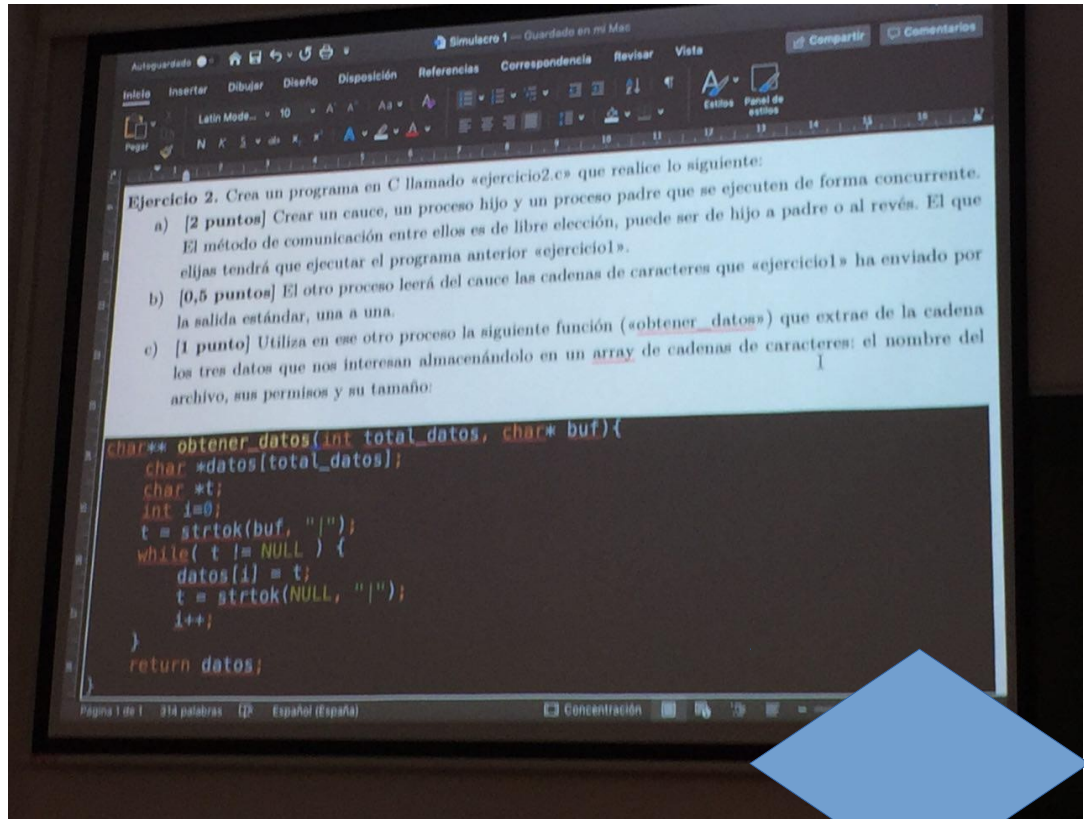
Ejercicio 2. Crea un programa en C llamado «ejercicio2.c» que realice lo siguiente:

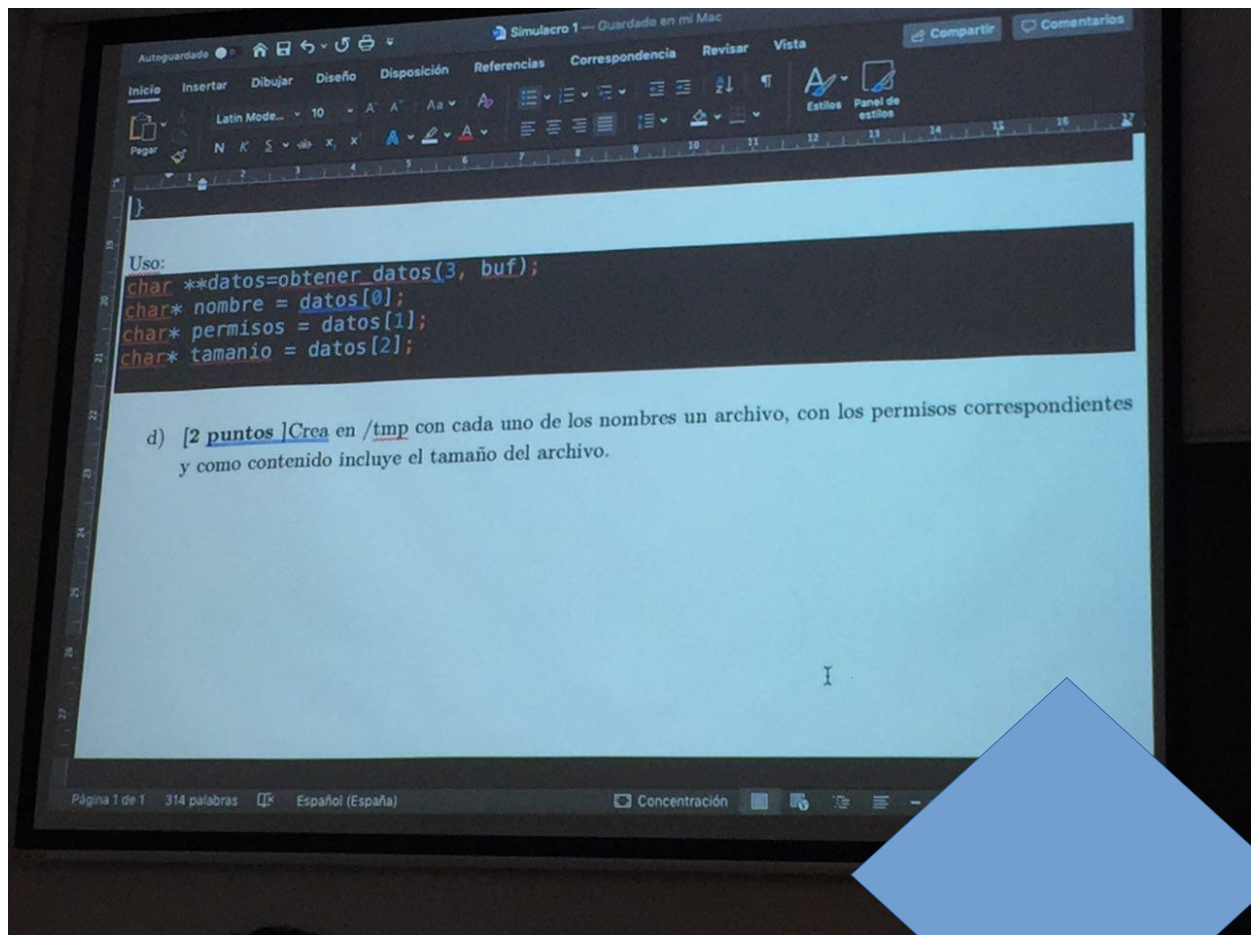
Página 1 de 1 314 palabras Español (España) Concentración 204 %



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.





Apellidos:

Nombre:

RECOMENDACIONES

- Esta prueba sirve para evaluar los conocimientos adquiridos hasta el momento.
 - No tengas miedo de preguntar algo que no comprendas; yo estoy aquí para ayudaros.
 - Ahora, olvida de los nervios y el estrés. ¡Sé que puedes hacerlo!
- #### INSTRUCCIONES:
- Acceda a su cuenta personal especificando el código **examenubu16** y levantando **Ubuntu 16.04 de 32 bits**.
 - No está permitido utilizar calculadora, móviles ni ningún dispositivo de comunicación o medio de comunicación.
 - Únicamente se accederá a Prado para acceder al trabajo definido para esta prueba.
 - Use únicamente las llamadas al sistema estudiadas en los guiones de prácticas del presente módulo.
 - Los programas deben contener únicamente lo que se pide, se penalizará si incluyen funcionalidad no requerida.
 - Programe la actuación adecuada ante las distintas situaciones de error que puedan ocurrir.

Ejercicio 1 [5 puntos]. La orden `ls -l` muestra en pantalla para cada hijo del directorio actual, su nombre y nº de i-nodo. Escriba un programa en C llamado `lsi.c` que tenga esta misma funcionalidad: para cada entrada del directorio de trabajo (un solo nivel) muestra su nombre y nº de i-nodo. Filtra esas entradas por:

- archivos de tipo regulares o directorios
- permisos 0777
- tamaño del archivo superior a 1byte
- longitud del nombre del archivo

no lseek

Ejercicio 2 [5 puntos]. Crea un programa en C llamado `cauces.c`. Utiliza un cauce para ejecutar `ls -l` (no ejecute el ejercicio anterior, ejecuta el comando). El resultado de ejecutar dicho comando será una cadena con este formato (i-nodo nombrearchivo):

```
690013 Documents
690015 Downloads
1826220 Dropbox
18006932 netflix.txt
```

Al final de cada línea existe un `"\n"`.

- Crea en `/tmp`, con permisos `777`, un archivo por cada nombre de archivo leído del comando. Como contenido guarda el i-nodo del archivo indicado.
- Suma el último número de cada i-nodo y muéstralo por consola. En el ejemplo: $3+5+0+2=10$
- Haz un recuento del número total de archivos devuelto por el comando. En nuestro ejemplo sería: 4.
- Guarda los resultados de a) y b) en un archivo `resultado.txt` de esta forma:

Suma inodos: 10
Total entradas: 4

b y c

Ejercicio extra [0,25 puntos]. Usa una sola macro con tres parámetros (el modo, el tamaño y la longitud del nombre) que filtre las entradas de `lsi.c` dando solución a todos los apartados, desde a hasta d.

Ejercicio extra [0,75 puntos]. Construye un nuevo programa en C llamado `netflix.c` al que le envíes por parámetros el nombre de una serie de Netflix. Deberá hacer lo mismo que `cauces.c` (cópialo por completo), pero que cuando el nombre de archivo sea `netflix.txt` cree (si no existe ya), un nuevo archivo regular en `/tmp` con el nombre `series_recomendadas.txt` y permisos `rw-r--r--`. Añada como contenido, siempre al final del archivo (use `\n` para separar) la nueva serie (parámetro `argv[1]`). Ejecútelo varias veces.