PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
Examen 2
(Primer semestre de 2019)

Horario 0781: prof. V. Khlebnikov
Horario 0782: prof. A. Bello R.

Duración: 3 horas
Nota: No se puede usar ningún material de consulta.
La presentación, la ortografía y la gramática influirán en la calificación.
El examen debe ser desarrollado en el cuadernillo usando lapicero.
Lo escrito con lápiz será considerado como borrador y NO será evaluado.
Puntaje total: 20 puntos

Pregunta 1 (10 puntos – 90 min.) (*From Wikipedia and Linux Mint man pages*) "In computer architecture, 64-bit computing is the use of processors that have datapath widths, integer size, and memory address widths of 64 bits (eight octets). Also, 64-bit computer architectures for central processing units (CPUs) and arithmetic logic units (ALUs) are those that are based on processor registers, address buses, or data buses of that size. From the software perspective, 64-bit computing means the use of code with 64-bit virtual memory addresses. In principle, a 64-bit microprocessor can address **… ?iBs** of memory. However, not all 64-bit instruction sets, and not all processors implementing those instruction sets, support a full 64-bit virtual memory addresses. The x86-64 architecture (also known as x64, x86_64, AMD64 and Intel 64) is the 64-bit version of the x86 instruction set, and allows 48 bits for virtual memory and, for any given processor, up to 52 bits for physical memory. These limits allow memory sizes of **… ?iB** and **… ?iB**, respectively."

a) (1 punto – 9 min.) Complete lo marcado tres veces en el texto con "**… ?iB**".

"In addition, the AMD specification requires that the most significant 16 bits of any virtual address, bits 48 through 63, must be copies of bit 47 (in a manner akin to sign extension). If this requirement is not met, the processor will raise an exception. Address complying with this rule are referred to as "canonical form." Canonical form addresses run from `0` through `00007FFF'FFFFFFFF`, and from `FFFF8000'00000000` through `FFFFFFFF'FFFFFFFF`. This feature eases later scalability to true 64-bit addressing. Many operating systems take the higher-addressed half of the address space (named kernel space) for themselves and leave the lower-addressed half (user space) for application code, user mode stacks, heaps, and other data regions. 64-bit Linux allows up to 128 TB of virtual address space for individual processes, and can address approximately 64 TB of physical memory, subject to processor and system limitations."

Using a Linux operating system, the command `uname` prints system information (kernel name, hostname, kernel release, kernel version, machine hardware name, processor type, hardware platform, operating system):

```
$ uname -a
Linux kaperna 4.15.0-54-generic #58-Ubuntu SMP Mon Jun 24 10:55:24 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux

$ getconf PAGESIZE
4096

$ free                                  # display amount of free and used memory in the system in kibibytes
              total       usado       libre  compartido búfer/caché  disponible
Memoria:   16216576     7820824     3057756     2219744     5337996     5839928
Swap:      33554428           0    33554428

$ ls -l /proc
total 0
dr-xr-xr-x  9 root             root                          0 jul  7 20:41 1
dr-xr-xr-x  9 root             root                          0 jul  7 20:41 10
dr-xr-xr-x  9 root             root                          0 jul  7 20:41 1010
dr-xr-xr-x  9 avahi            avahi                         0 jul  7 20:42 1034
dr-xr-xr-x  9 ntp              ntp                           0 jul  7 20:41 1050
dr-xr-xr-x  9 root             root                          0 jul  7 20:41 11
dr-xr-xr-x  9 root             root                          0 jul  7 20:41 1127
...
dr-xr-xr-x  9 daemon           daemon                        0 jul  7 20:41 951
```

```
dr-xr-xr-x  2 root            root                          0 jul  7 20:41 acpi
dr-xr-xr-x  6 root            root                          0 jul  7 20:41 asound
-r--r--r--  1 root            root                          0 jul  9 01:20 buddyinfo
dr-xr-xr-x  4 root            root                          0 jul  7 20:41 bus
-r--r--r--  1 root            root                          0 jul  9 01:20 cgroups
-r--r--r--  1 root            root                          0 jul  9 01:20 cmdline
-r--r--r--  1 root            root                          0 jul  9 01:20 consoles
-r--r--r--  1 root            root                          0 jul  9 01:20 cpuinfo
...
-r--------  1 root            root                          0 jul  9 01:20 vmallocinfo
-r--r--r--  1 root            root                          0 jul  9 01:20 vmstat
-r--r--r--  1 root            root                          0 jul  9 01:20 zoneinfo
```

“The `proc` filesystem is a pseudo-filesystem which provides an interface to kernel data structures. It is commonly mounted at `/proc`. Most of the files in the `proc` filesystem are read-only, but some files are writable, allowing kernel variables to be changed.

The /proc/[pid] is a numerical subdirectory for each running process; the subdirectory is named by the process ID. ”

```
$ ls -l /proc/9212
total 0
dr-xr-xr-x 2 vk vk 0 jul  9 01:36 attr
-rw-r--r-- 1 vk vk 0 jul  9 01:36 autogroup
-r-------- 1 vk vk 0 jul  9 01:36 auxv
-r--r--r-- 1 vk vk 0 jul  9 01:36 cgroup
--w------- 1 vk vk 0 jul  9 01:36 clear_refs
-r--r--r-- 1 vk vk 0 jul  8 23:09 cmdline
-rw-r--r-- 1 vk vk 0 jul  9 01:36 comm
-rw-r--r-- 1 vk vk 0 jul  9 01:36 coredump_filter
-r--r--r-- 1 vk vk 0 jul  9 01:36 cpuset
lrwxrwxrwx 1 vk vk 0 jul  9 01:36 cwd -> /home/vk
-r-------- 1 vk vk 0 jul  9 01:36 environ
lrwxrwxrwx 1 vk vk 0 jul  8 23:09 exe -> /usr/lib/libreoffice/program/soffice.bin
dr-x------ 2 vk vk 0 jul  9 01:36 fd
dr-x------ 2 vk vk 0 jul  9 01:36 fdinfo
-rw-r--r-- 1 vk vk 0 jul  9 01:36 gid_map
-r-------- 1 vk vk 0 jul  9 01:36 io
-r--r--r-- 1 vk vk 0 jul  9 01:36 limits
-rw-r--r-- 1 vk vk 0 jul  9 01:36 loginuid
dr-x------ 2 vk vk 0 jul  9 01:36 map_files
-r--r--r-- 1 vk vk 0 jul  9 01:36 maps
-rw------- 1 vk vk 0 jul  9 01:36 mem
-r--r--r-- 1 vk vk 0 jul  9 01:36 mountinfo
-r--r--r-- 1 vk vk 0 jul  9 01:36 mounts
-r-------- 1 vk vk 0 jul  9 01:36 mountstats
dr-xr-xr-x 5 vk vk 0 jul  9 01:36 net
dr-x--x--x 2 vk vk 0 jul  9 01:36 ns
-r--r--r-- 1 vk vk 0 jul  9 01:36 numa_maps
-rw-r--r-- 1 vk vk 0 jul  9 01:36 oom_adj
-r--r--r-- 1 vk vk 0 jul  9 01:36 oom_score
-rw-r--r-- 1 vk vk 0 jul  9 01:36 oom_score_adj
-r-------- 1 vk vk 0 jul  9 01:36 pagemap
-r-------- 1 vk vk 0 jul  9 01:36 patch_state
-r-------- 1 vk vk 0 jul  9 01:36 personality
-rw-r--r-- 1 vk vk 0 jul  9 01:36 projid_map
lrwxrwxrwx 1 vk vk 0 jul  9 01:36 root -> /
-rw-r--r-- 1 vk vk 0 jul  9 01:36 sched
-r--r--r-- 1 vk vk 0 jul  9 01:36 schedstat
-r--r--r-- 1 vk vk 0 jul  9 01:36 sessionid
-rw-r--r-- 1 vk vk 0 jul  9 01:36 setgroups
-r--r--r-- 1 vk vk 0 jul  9 01:36 smaps
-r--r--r-- 1 vk vk 0 jul  9 01:36 smaps_rollup
-r-------- 1 vk vk 0 jul  9 01:36 stack
-r--r--r-- 1 vk vk 0 jul  9 00:52 stat
-r--r--r-- 1 vk vk 0 jul  9 01:31 statm
-r--r--r-- 1 vk vk 0 jul  9 00:52 status
-r-------- 1 vk vk 0 jul  9 01:36 syscall
dr-xr-xr-x 7 vk vk 0 jul  9 01:36 task
-r--r--r-- 1 vk vk 0 jul  9 01:36 timers
-rw-rw-rw- 1 vk vk 0 jul  9 01:36 timerslack_ns
-rw-r--r-- 1 vk vk 0 jul  9 01:36 uid_map
-r--r--r-- 1 vk vk 0 jul  9 01:36 wchan
```

The file `/proc/[pid]/maps` containing the currently mapped memory regions and their access permissions. For example, for a user program `pagemap2` we have:

```
address                   perms offset  dev   inode                     pathname
558cf5c45000-558cf5c47000 r-xp 00000000 08:08 13505283                  /home/vk/clases/so/progs/pagemap2
558cf5e46000-558cf5e47000 r--p 00001000 08:08 13505283                  /home/vk/clases/so/progs/pagemap2
558cf5e47000-558cf5e48000 rw-p 00002000 08:08 13505283                  /home/vk/clases/so/progs/pagemap2
```

```
7fd04d7bc000-7fd04d9a3000 r-xp 00000000 08:07 3670237                    /lib/x86_64-linux-gnu/libc-2.27.so
7fd04d9a3000-7fd04dba3000 ---p 001e7000 08:07 3670237                    /lib/x86_64-linux-gnu/libc-2.27.so
7fd04dba3000-7fd04dba7000 r--p 001e7000 08:07 3670237                    /lib/x86_64-linux-gnu/libc-2.27.so
7fd04dba7000-7fd04dba9000 rw-p 001eb000 08:07 3670237                    /lib/x86_64-linux-gnu/libc-2.27.so
7fd04dba9000-7fd04dbad000 rw-p 00000000 00:00 0
7fd04dbad000-7fd04dbd4000 r-xp 00000000 08:07 3670229                    /lib/x86_64-linux-gnu/ld-2.27.so
7fd04dda9000-7fd04ddab000 rw-p 00000000 00:00 0        /* ld es dynamic linker/loader (comentario de vk) */
7fd04ddd4000-7fd04ddd5000 r--p 00027000 08:07 3670229                    /lib/x86_64-linux-gnu/ld-2.27.so
7fd04ddd5000-7fd04ddd6000 rw-p 00028000 08:07 3670229                    /lib/x86_64-linux-gnu/ld-2.27.so
7fd04ddd6000-7fd04ddd7000 rw-p 00000000 00:00 0
7ffec277e000-7ffec279f000 rw-p 00000000 00:00 0                          [stack]
7ffec279f000-7ffec27a2000 r--p 00000000 00:00 0                          [vvar]
7ffec27a2000-7ffec27a4000 r-xp 00000000 00:00 0                          [vdso]
```

b) (3 puntos – 27 min.) ¿Cuántas páginas virtuales ocupa todo el programa `pagemap2` en su espacio de direcciones virtuales? Realice todo el cálculo con los valores en hexadecimal, y el resultado final convierta al decimal.

The file `/proc/[pid]/pagemap` shows the mapping of each of the process's virtual pages into physical page frames or swap area. It contains one 64-bit value for each virtual page, with the bits set as follows:

63	If set, the page is present in RAM.
62	If set, the page is in swap space.
61	The page is a file-mapped page or a shared anonymous page.
60-57	Zero
56	The page is exclusively mapped.
55	PTE (page table entry) is soft-dirty.
54-0	If the page is present in RAM (bit 63), then these bits provide the page frame number. If the page is present in swap (bit 62), then bits 4-0 give the swap type, and bits 54-5 encode the swap offset.

c) (1 punto – 9 min.) ¿Cuál es el tamaño del seudoarchivo `/proc/[pid]/pagemap`?

d) (1 punto – 9 min.) Por ser tan grande el archivo, su lectura byte por byte (con `getc()`) toma casi una media hora. Pero con `lseek()` se puede saltar directamente a la entrada del archivo correspondiente a la primera página virtual usada por el programa `pagemap2` que analiza su propio `pagemap`. ¿Cómo se calcula el `offset` para este `lseek()` y cuál será?

e) (1 punto – 9 min.) ¿Cuántos marcos de página hay en el sistema que estamos usando? ¿Cuál es el rango de sus números (en hex)?

f) (1 punto – 9 min.) Interprete las entradas del archivo `/proc/[pid]/pagemap`, correspondientes a las 3 primeras páginas virtuales usadas durante la ejecución del programa `pagemap2`. Estas entradas son las siguientes:

```
0xa1800000002635c1
0xa1800000001e32e4
0x81800000001731ee
```

g) (1 punto – 9 min.) Durante la ejecución del programa `pagemap2`, que lee y analiza su propio archivo `/proc/[pid]/pagemap`, se descubre que solamente 402 páginas virtuales están mapeadas en RAM y no hay ninguna página virtual en *swap*. Haga su suposición de por qué las páginas restantes (¿cuántas son?) no están presentes.

h) (1 punto – 9 min.) Durante la ejecución del programa `pagemap2`, que duró 24 minutos, sucedieron solamente 221 fallos de página. Esta cantidad es un poco más que la mitad de las páginas virtuales del programa mapeadas en RAM. ¿Por qué las otras páginas virtuales ya estaban en RAM antes que comenzó la ejecución del programa?

<u>Pregunta 2</u> (4 puntos – 36 min.) En una terminal de Linux Mint 19.1 se realiza el siguiente procedimiento:

```
$ dd if=/dev/zero of=./disquete.msdos bs=1024 count=1440
1440+0 registros leídos
1440+0 registros escritos
1474560 bytes (1,5 MB, 1,4 MiB) copied, 0,0029429 s, 501 MB/s

$ /sbin/mkfs.fat -n SISTEMASOPERATIVOS -s 1 disquete.msdos
mkfs.fat 4.1 (2017-01-24)

$ fatcat disquete.msdos -i
FAT Filesystem information

Filesystem type: FAT12
OEM name: mkfs.fat
Total sectors: 2880
Total data clusters: 3072
Data size: 1572864 (1.5M)
```

```
Disk size: 1474560 (1.40625M)
Bytes per sector: 512
Sectors per cluster: 1
Bytes per cluster: 512
Reserved sectors: 1
Root entries: 224
Root clusters: 14
Sectors per FAT: 9
Fat size: 4608 (4.5K)
FAT1 start address: 0000000000000200
FAT2 start address: 0000000000001400
Data start address: 0000000000002600
Root directory cluster: 0
Disk label: SISTEMASOPE

Free clusters: 3070/3072 (99.9349%)
Free space: 1571840 (1.49902M)
Used space: 1024 (1K)

$ udisksctl loop-setup -f "/home/alulab/Documentos/disquete.msdos"
Mapped file /home/alulab/Documentos/disquete.msdos as /dev/loop0.

$ seq -f "File%03g"  1 100 | while read f1 && read f2;do for i in $(seq 1 117);do \
./fragm /media/alulab/SISTEMASOPE/$f1; ./fragm /media/alulab/SISTEMASOPE/$f2;done;done


$ ls /media/alulab/SISTEMASOPE/
File001  File011  File021  File031  File041  File051  File061  File071  File081  File091
File002  File012  File022  File032  File042  File052  File062  File072  File082  File092
File003  File013  File023  File033  File043  File053  File063  File073  File083  File093
File004  File014  File024  File034  File044  File054  File064  File074  File084  File094
File005  File015  File025  File035  File045  File055  File065  File075  File085  File095
File006  File016  File026  File036  File046  File056  File066  File076  File086  File096
File007  File017  File027  File037  File047  File057  File067  File077  File087  File097
File008  File018  File028  File038  File048  File058  File068  File078  File088  File098
File009  File019  File029  File039  File049  File059  File069  File079  File089  File099
File010  File020  File030  File040  File050  File060  File070  File080  File090  File100

$ udisksctl unmount -b /dev/loop0
Unmounted /dev/loop0.

$ fatcat disquete.msdos -i
FAT Filesystem information

Filesystem type: FAT12
OEM name: mkfs.fat
Total sectors: 2880
Total data clusters: 3072
Data size: 1572864 (1.5M)
Disk size: 1474560 (1.40625M)
Bytes per sector: 512
Sectors per cluster: 1
Bytes per cluster: 512
Reserved sectors: 1
Root entries: 224
Root clusters: 14
Sectors per FAT: 9
Fat size: 4608 (4.5K)
FAT1 start address: 0000000000000200
FAT2 start address: 0000000000001400
Data start address: 0000000000002600
Root directory cluster: 0
Disk label: SISTEMASOPE

Free clusters: 223/3072 (7.25911%)
Free space: 114176 (111.5K)
Used space: 1458688 (1.39111M)
```

$ cat ./fragm.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>


int main(int narg, char *argv[]){
        char registro[128];
        int fd;
        unsigned char i;

        for(i=0;i<128;i++) registro[i] = i;
```

```
        fd=open(argv[1],O_WRONLY | O_CREAT,0660);
        lseek(fd,0,SEEK_END);
        write(fd, registro, sizeof(registro));
        close(fd);
        return 0;
}
```

a) (2 puntos – 18 min.) ¿Qué archivos tienen el mismo tamaño?
b) (1 punto – 9 min.) ¿Qué archivos tienen diferente tamaño y cuál es su tamaño (en sectores)?
c) (1 punto – 9 min.) ¿Qué archivos tienen tamaño 0?

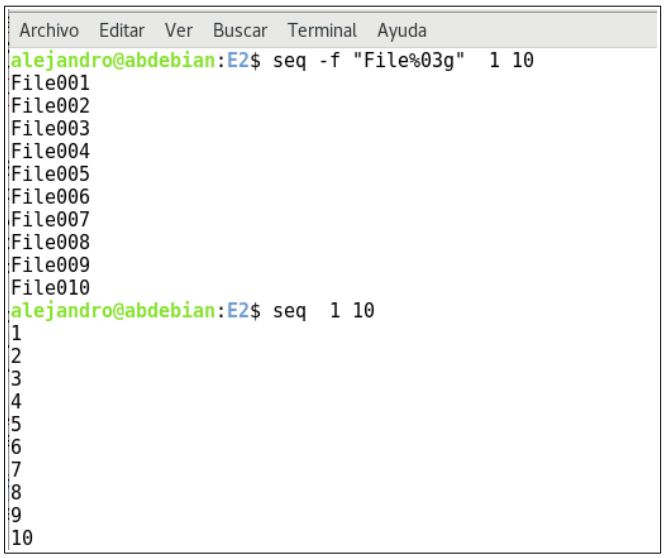
Pregunta 3 (6 puntos – 54 min.) En la misma terminal se realiza un procedimiento análogo a la pregunta anterior, pero en este caso el formato del sistema de archivos es ext2.

```
$ dd if=/dev/zero of=./disquete.ext2 bs=1024 count=1440
1440+0 registros leídos
1440+0 registros escritos
1474560 bytes (1,5 MB, 1,4 MiB) copied, 0,0029429 s, 501 MB/s

$ /sbin/mkfs.ext2 -L "SistemasOperativos" disquete.ext2
Atención: la etiqueta es demasiado larga; se trunca a 'SistemasOperativ'
mke2fs 1.43.4 (31-Jan-2017)
Descartando los bloques del dispositivo: hecho
Se está creando un sistema de ficheros con 1440 bloques de 1k y 184 nodos-i

Reservando las tablas de grupo: hecho
Escribiendo las tablas de nodos-i: hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho

$ /sbin/dumpe2fs disquete.ext2
dumpe2fs 1.43.4 (31-Jan-2017)
Filesystem volume name:   SistemasOperativ
Last mounted on:          <not available>
Filesystem UUID:          23f013df-e5d8-45cf-abfc-6c0edd4290e9
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      ext_attr resize_inode dir_index filetype sparse_super large_file
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              184
Block count:              1440
Reserved block count:     72
Free blocks:              1405
Free inodes:              174
First block:              1
Block size:               1024
Fragment size:            1024
Reserved GDT blocks:      5
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         184
Inode blocks per group:   23
Filesystem created:       Tue Jul  9 17:42:03 2019
Last mount time:          n/a
Last write time:          Tue Jul  9 17:42:03 2019
Mount count:              0
Maximum mount count:      -1
Last checked:             Tue Jul  9 17:42:03 2019
Check interval:           0 (<none>)
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Default directory hash:   half_md4
Directory Hash Seed:      7b6b134d-0114-4249-aa85-184c90833f8d


Grupo 0: (Bloques 1-1439)
  Superbloque primario en 1, descriptores de grupo en 2-2
  Se reservaron los bloques GDT en 3-7
  Mapa de bits de bloques en 8 (+7)
  Mapa de bits de nodos-i en 9 (+8)
  Tabla de nodos-i en 10-32 (+9)
  1405 bloques libres, 174 nodos-i libres, 2 directorios
  Bloques libres: 34-45, 47-1439
  Nodos-i libres: 11-184
```

```
$ udisksctl loop-setup -f "/home/alulab/Documentos/disquete.ext2"
Mapped file /home/alulab/Documentos/disquete.ext2 as /dev/loop0.


$ seq -f "File%03g"  1 100 | while read f1 && read f2;do for i in $(seq 1 117);do\
./fragm /media/alulab/SistemasOperativ/$f1; ./fragm /media/alulab/SistemasOperativ/$f2;done;done



$ ls /media/alejandro/SistemasOperativ/
File001  File011  File021  File031  File041  File051  File061  File071  File081  File091
File002  File012  File022  File032  File042  File052  File062  File072  File082  File092
File003  File013  File023  File033  File043  File053  File063  File073  File083  File093
File004  File014  File024  File034  File044  File054  File064  File074  File084  File094
File005  File015  File025  File035  File045  File055  File065  File075  File085  File095
File006  File016  File026  File036  File046  File056  File066  File076  File086  File096
File007  File017  File027  File037  File047  File057  File067  File077  File087  File097
File008  File018  File028  File038  File048  File058  File068  File078  File088  File098
File009  File019  File029  File039  File049  File059  File069  File079  File089  File099
File010  File020  File030  File040  File050  File060  File070  File080  File090  File100

$ udisksctl unmount -b /dev/loop0
Unmounted /dev/loop0.

$ /sbin/dumpe2fs disquete.ext2
dumpe2fs 1.43.4 (31-Jan-2017)
Filesystem volume name:   SistemasOperativ
Last mounted on:          /media/alejandro/SistemasOperativ
Filesystem UUID:          23f013df-e5d8-45cf-abfc-6c0edd4290e9
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      ext_attr resize_inode dir_index filetype sparse_super large_file
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              184
Block count:              1440
Reserved block count:     72
Free blocks:              72
Free inodes:              74
First block:              1
Block size:               1024
Fragment size:            1024
Reserved GDT blocks:      5
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         184
Inode blocks per group:   23
Filesystem created:       Tue Jul  9 17:42:03 2019
Last mount time:          Tue Jul  9 17:45:41 2019
Last write time:          Tue Jul  9 17:54:46 2019
Mount count:              1
Maximum mount count:      -1
Last checked:             Tue Jul  9 17:42:03 2019
Check interval:           0 (<none>)
Lifetime writes:          1363 kB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Default directory hash:   half_md4
Directory Hash Seed:      7b6b134d-0114-4249-aa85-184c90833f8d


Grupo 0: (Bloques 1-1439)
  Superbloque primario en 1, descriptores de grupo en 2-2
  Se reservaron los bloques GDT en 3-7
  Mapa de bits de bloques en 8 (+7)
  Mapa de bits de nodos-i en 9 (+8)
  Tabla de nodos-i en 10-32 (+9)
  72 bloques libres, 74 nodos-i libres, 1 directorios
  Bloques libres: 484-512, 1002-1024, 1405-1424
  Nodos-i libres: 111-184
```

a) (4 puntos – 36 min.) ¿Qué archivos tienen el mismo tamaño?
b) (1 punto – 9 min.) ¿Qué archivos tienen diferente tamaño y cuál es su tamaño (en sectores)?
c) (1 punto – 9 min.) ¿Qué archivos tienen tamaño 0?

ANEXO



Preparado por AB(2,3) y VK(1) con LibreOffice Writer
en Linux Mint 19.1 Tessa

Profesores del curso: (0781) V. Khlebnikov
(0782) A. Bello R.

Pando, 10 de julio de 2019