SURNAME		NAME		Group
ID		Signature		

- **Keep the exam sheets stapled.**
- **Write your answer inside the reserved space.**
- **Use clear and understandable writing. Answer briefly and precisely.**
- **The exam has 7 questions, everyone has its score specified.**

---

1. Libraries are files that contain code functions and can be linked statically or dynamically. Indicate by placing an X, which of the following characteristics correspond to one or another type of linking or both: (Note: An error voids a correct answer). **(1,0 point)**

1	**Feature**	**Static**	**Dynamic**
	It generates executable files of smaller size		X
	It generates executable files containing the own process code and the libraries code	X	
	In the process memory map they appear independent regions whose support is the code file for each library linked		X
	Several processes can share the library code after being allocated in main memory		X
	Whenever the OS updates a library, the executable files that use that library have to be rebuild	X	
	Library linking is done at runtime and this can cause execution delay		X
	It can generate multiple copies of the same library on main memory	X	
	On the memory map the code of libraries has support on the code file of the process itself	X	

---

2. Indicate how the MMU (Memory Management Unit) solves process relocation at runtime on contiguous allocation and what information it needs to carry it out: **(0.5 points)**

2	The MMU translates logical addresses (LA) to physical (PA), it needs to know the main memory addresses where the process is allocated. Every time a process changes its memory allocation on main memory the information about the new location has to be updated on the MMU to perform translations correctly. With contiguous allocation the MMU uses one base and one limit register per process with the PA from which the process has been located, on the base register, and the process size on the limit register. The base register has to be updated every time there is a process relocation.

3. A system with pure paging (without virtual memory), 32-bit logical addresses, equipped with a 64GBytes of main memory, uses 20 bits for the page id and 6 bytes for each page descriptor. Explain your answer to the following sections:

(1.5 points =0.5+0.25+0.25+0.5)

3

a) Formats for logical and physical addresses, telling the name and number of bits for every address field

Logical address 32 bits → 32 - 20 = 12 bits for the offset
64 GBytes = 2^6 2^{30} Bytes = 2^{36} Bytes → Physical address 36 bits
→ 36 - 12 = 24 bits for the frame id

Logical address

Page id (20 bits)	*Offset (12 bits)*

b31 *b12 b11* *b0*

Physical address

Frame id (24 bits)	*Offset(12 bits)*

b35 *b12 b11* *b0*

b) Page table size of a process

If logical addresses are 32 bits, with 20 bits for the page id then
→ 2^{20} is the maximum number of pages a proces s can have
The page table needs a descriptor por page so:
→ 2^{20} páginas * 6 Bytes → 6 MBytes if the pag e table size

c) Maximum multiprogramming level considering processes of 4 GByte

The maximum multiprogramming level is the number of processes that can be allocated in memory simultaneously. Due to pure paging (no virtual memory) processes have to be fully allocated in main memory to run. The maximum size of a process is given by the size of the logical space 4GByte = 2^{32}, while the number of processes that can be placed in memory is given by the division between physical space and the size of a process then:
64 GBytes / 2^{32} = 2^{36} /2^{32} = 2^4 = 16 processes

d) Considering two levels paging, with 256 second-level descriptors, obtain the formats of logical and physical addresses indicating the name and number of bits for every address field

256 descriptors = 2^8 so 8 bits to address them, as we have 20 bits for the page ido then:
→ 20 - 8 = 12 for the first level, the physical address doesn't change

Logical address

1st level (12 bits)	*2nd level (8 bits)*	*Offset (12 bits)*

b31 *b20 b19* *b12 b11* *b0*

Physical address

Frame id (24 bits)	*Offset(12 bits)*

b35 *b12 b11* *b0*

4. On a system with demand paging (virtual memory) and local replacement policy, the maximum size of a logical process is 4K pages and the page size is 64Kbytes. The following table contains all the information about P and Q processes at time t = 170:

Information about P and Q processes at t = 170							
Process	**Frame**	**Page**	**Load instant**	**Last reference instant**	**Bit R (reference)**	**Bit M (modified)**	**Bits RWX**
P	0x4A	0xC72	160	161	1	0	101
P	0x4B	0xC71	120	140	1	1	101
P	0x4C	0xA70	36	152	0	1	110
P	0x4D	0xA73	30	163	1	1	110
Q	0x4E	0xA70	40	167	1	0	101
Q	0x4F	0xA73	42	142	0	1	110

Based on the information provided on the table, answer each section: **(2 points=0.25+0.5+0.5+0.75)**

4

a) Indicate which pages of P and Q have the valid bit set to 0 and enter the content of the page descriptors for pages with valid bit set to 1

In the page table all pages have the validity bit to 0 except the ones loaded into main memory. The descriptor of each page will have the frame id where it is allocated and control bits (V, RWX, etc)

Process P	**Page**	**Descriptor**	**Process Q**	**Page**	**Descriptor**
	0xA70	0x4C v=1		0xA70	0x4E v=1
	0xA73	0x4D v=1		0xA73	0x4F v=1
	0xC71	0x4B v=1			
	0xC72	0x4A v=1			

b) From the table obtain the logical address corresponding to the physical address 0x4D4AB1

Physical address 0x4D4AB1 corresponds to process P as it is page A73 of this process which is located on the frame 4D. It can't correspond to Q.

Physical address of P -> Logical address	*Physical address of Q -> Logical address*
0x4D4AB1 ---> 0xA734AB1	0x4D4AB1 ---> BELONGS TO P

c) At time t = 171 the CPU sends P's logical address 0xB95603A and at time t = 172 the CPU sends Q's logical address 0xB95603A. Compute the corresponding physical addresses if the replacement policy is **second chance**:

Logical space 4K pages → 12 bits for page id → 3 hex digits
Page size is 64 KBytes → 16 bits for offset -> 4 hex digits
2nd chance → The victim is the first found with reference bit = 0
For P -> page A70 allocated on frame 4C
For Q -> page A70 allocated on frame 4F

Logical address of P -> Physical address	*Logical address of Q -> Physical address*
0xB95603A ---> 4C603A	0xB95603A ---> 4F603A

d) Content evolution of the frames involved if the replacement policy is **LRU (least recently used)**, considering that from t = 171 the CPU sends P addresses with the following reference string : 0xA71, 0xB40, 0xC72, 0xB51

Frame	t = 170 (Start)	t=171 ->A71	t=172->B40	t=173->C72	t=174->B51	
0x4A	0xC72(161)	0xC72(161)	0xC72(161)	0xC72(173)	0xC72(173)	
0x4B	0xC71(140)	**0xA71(171)**	0xA71(171)	0xA71(171)	0xA71(171)	
0x4C	0xA70(152)	0xA70(152)	**0xB40(172)**	0xB40(172)	0xB40(172)	
0x4D	0xA73(163)	0xA73(163)	0xA73(163)	0xA73(163)	**0xB51(174)**	

5. When running the following code in C three processes are created:

```
{int fd_pipe[2], fd; /* pipe descriptor, regular file */
 int pid;
 /*** Initial table *************************/
  close(1);
  fd=open("result",O_WRONLY |O_CREAT|O_TRUNC,0666);
  dup2(2,3);
  fd=open("datos",O_RDONLY);
  close(0);close(2);
  pipe(fd_pipe);
  pid=fork();
  if (pid==0)
       { close(fd);
         dup2(3,2);
         /*** P2 table *********************/
         execlp("/usr/bin/wc", "wc", "-l",NULL);
       }
  pid=fork();
  if (pid==0)
       { dup2(2,1); dup2(fd,0);  dup2(3,2);
         /*** P3 table ******************/
         execlp("/bin/cat", "cat", NULL);
       }
   close(0); close(2);
         /*** P1 table ******************/
  while(pid != wait(&status));
}
```

	Initial table
0	STDIN
1	STDOUT
2	STDERR
3	
4	

(1.6 points=0.4+1.2)

5

a) Describe the relationship between processes P1, P2 and P3, and draw the communication scheme established

Three process are created: P1 (parent) and P2 and P3 (children of P1) . Children inherit the parent's pipe. The communication established is: P3 reads file "data" and writes into the pipe, P2 reads from the pipe and writes on file "result"

b) Obtain the descriptor table content for every process involved at the marks inserted on the code as /*** … table … ***/

	P1 table
0	
1	"result"
2	
3	STDERR
4	"datos"
5	

	P2 table
0	"fd_pipe[0]"
1	"result"
2	STDERR
3	STDERR
4	
5	

	P3 table
0	"datos"
1	"fd_pipe[1]"
2	STDERR
3	STDERR
4	"datos"
5	

6. Given the following directory listing on a POSIX system:

```
permissions links   user   group   size       date          name
drwxr-xr-x   2      sterr    fso     4096    dec  9 2016    .
drwxrwxr-x   8      sterr    fso     4096    sep 10 2016    ..
-r-xr-sr-x   1      sterr    fso  1139706    dec  9 2016    cp1
-r-sr-xr-x   1      sterr    fso  1139706    dec  9 2016    cp2
-r-xr-xr-x   1      sterr    fso  1139706    dec  9 2016    cp3
-r--r-----   1      sterr    fso     9706    dec  9 2016    f1
-r--rw-rw-   1      sterr    fso     4157    dec  9 2016    f2
-rw-r--r--   1      sterr    fso      222    dec  9 2016    f4
```

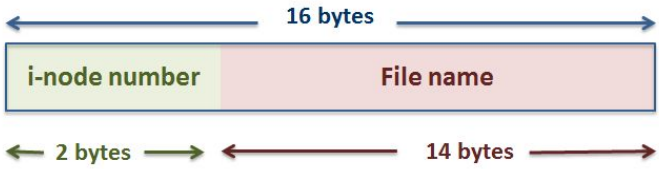
(1.4 points)

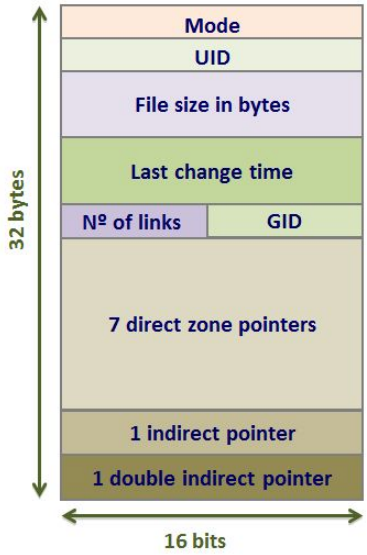
6

Considering that programs cp1, cp2 and cp3 are three identical copies of the cp system command that copies the contents of the file on the first argument into another one which name is indicated as the second argument. That is, "cp1 a b" copies the contents of file "a" into file "b", if "b" does not exist it is created and the copied. Fill the table indicating whether the command works or not, the EUID and EGID (effective process UID and GID when the command is being executed) and in case of error, which is the permission that fails.

UID, GID	Command	Does it work?	EUID, EGID	In case of error explain
pepe, fso	cp2 f1 f2	No	sterr, fso	Writing in f2
sterr, etc	cp1 f1 f4	Yes	sterr, fso	-
ana, etc	cp1 f1 f2	Yes	ana, fso	-
ana, etc	cp1 f1 f5	No	ana, fso	Writing in .
ana, etc	cp2 f1 f5	Yes	sterr, etc	-
ana, etc	cp3 f1 f2	No	ana, etc	Reading f1
ana, etc	cp3 f4 f2	Yes	ana, etc	-

7. The following figures refer to the sizes and structures of the elements of a MINIX file system used to format a disk partition. The block size is 1KByte and 1 zone = 1 block. Note that all fields in the i-node are 16 bits except "No. of links" and GID that are 8 bits. The directory entry format is:





(2,0 points)

a) Calculate the partition size given that it has been formatted for the maximum number of zones and to have the maximum number of i-nodes

Since zone pointers are 16-bit, we can have up to 2^{16} zones = 64K zones, as each zone is 1KB, so the maximum partition size is 64K x 1KB = 64MB

b) Maximum number of i-nodes

Since the i-node number is 16-bit, we can have up to 2^{16} i-nodes = 64K files

Answer the following sections considering a MINIX file system with the parameters described above and having a partition size of 32MBytes (Megabytes) and 32K i-nodes :

c) Number of blocks required for the i-node bit map, the zone bit map and the i-nodes

32MB / 1KB = 32K zones, as we have the same number of i-nodes than zones (32K) then the size of both bitmap will be the same 32K / 8Kbits = 4 blocks. The number of blocks for i-nodes will 32K x 32B / 1KB = 1024 blocks

d) Free disk space after formatting (including the creation of the root directory)

The number of zones is 32K-1-1-4-4-1024 = 31734 zones. The empty root directory occupies one zone, so the free space is 31733 KB

e) Maximum number of physical links that a file can have

The number of links field is 8 bit, then the maximum number of physical links is 2^8 - 1 = 255

f) Maximum number of symbolic links that a file can have

Every symbolic link is a file so it requires an i-node. The maximum number of i-nodes is 32K so this is an upper bound for the number of symbolic links

g) Maximum number of directories that a directory can contain

Every directory contains ".” that points to it and ".." that points to its parent (except the root directory), so the number of links to a directory is: the directory entry corresponding to the directory + all the directory entries contained + directory entry ".". Because the number of links field is 8 bithe answer is:

255 - 2 = 253 (this is true also for the root directory because "." and ".." point both to i-node 1)

h) Number of zones occupied by 10 files of 100 Bytes

A file of size 100 byte needs only one data zone, so 10 files take 10 zones

i) Number of zones occupied by 10 files of 100 KBytes

A file of size 100 KB needs 100 data zones so the indirect pointer is required, then it will occupy 101 areas. Finally 10 files will take 1010