Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

Part 4: Memory management

Seminar 12

Virtual memory exercises (II)

fSO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA

- **Exercise S10-1: 2nd chance replacement algorithm**
  - Exercise S10-1.1: 2nd chance basic
  - Exercise S10-1.2: 2nd chance complete
- Exercise S10-2: Working set
  - Exercise S10-2.1: Getting the working set

Exercise S10-1.1

- In a **demand paging** system with **global replacement** policy, **5 frames** are allocated to all **processes**. Pages (and frames) have a size of **16K pages**. The initial state of the system and the page sequence generated by the CPU is detailed below.

Page Sequence	**P1,0**	**P1,3**	**P2,3**	**P2,4**	**P1,3**	**P2,1**	**P1,4**	**P1,0**	**P2,2**
Time	21	22	23	24	25	26	27	28	29

Initial frame allocation	
0	P1,1
1	P2,2
2	P1,2
3	P1,3
4	--

Obtain what pages are allocated on every frame when the next global scope algorithms are applied

a) LRU based on Counters considering the initial counters state
b) LRU based on stack considering the initial stack state
c) LRU based in bit reference and 2nd oportunity considering the initial bit state

Exercise S10-1.1

- In a **demand paging** system with **local replacement** policy, **5 frames** are allocated to all **processes**. Pages (and frames) have a size of **16K pages**. The initial state of the system and the page sequence generated by the CPU is detailed below.
- Obtain what pages are allocated on every frame when the next global scope algorithms are applied

a) LRU based on Counters

b) LRU based on stack

c) LRU based in bit reference and 2nd opportunity.

	Inicio	P1,0	P1,3	P2,3	P2,4	P1,3	P2,1	P1,4	P1,0	P2,2
tiempo	20	21	22	23	24	25	26	27	28	29
0	P1,1	P1,1	P1,1	P2,3	P2,3	P2,3	P2,3	P2,3	P1,0	P1,0
1	P2,2	P2,2	P2,2	P2,2	P2,4	P2,4	P2,4	P2,4	P2,4	P2,2
2	P1,2	P1,2	P1,2	P1,2	P1,2	P1,2	P2,1	P2,1	P2,1	P2,1
3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3
4	--	P1,0	P1,0	P1,0	P1,0	P1,0	P1,0	P1,4	P1,4	P1,4
0	10	10	10	23	23	23	23	23	28	28
1	11	11	11	11	24	24	24	24	24	29
2	12	12	12	12	12	12	26	26	26	26
3	13	13	22	22	22	25	25	25	25	25
4	--	21	21	21	21	21	21	27	27	27

Counter of page in frame

Faults: 7 Replacements: 6

Exercise S10-1.1

- In a **demand paging** system with **local replacement** policy, **5 frames** are allocated to all **processes**. Pages (and frames) have a size of **16K pages**. The initial state of the system and the page sequence generated by the CPU is detailed below.
- Obtain what pages are allocated on every frame when the next global scope algorithms are applied

a) LRU based on Counters

b) LRU based on stack

c) LRU based in bit reference and 2nd opportunity.

	Inicio	P1,0	P1,3	P2,3	P2,4	P1,3	P2,1	P1,4	P1,0	P2,2
tiempo	20	21	22	23	24	25	26	27	28	29
0	P1,1	P1,1	P1,1	P2,3	P2,3	P2,3	P2,3	P2,3	P1,0	P1,0
1	P2,2	P2,2	P2,2	P2,2	P2,4	P2,4	P2,4	P2,4	P2,4	P2,2
2	P1,2	P1,2	P1,2	P1,2	P1,2	P1,2	P2,1	P2,1	P2,1	P2,1
3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3
4	--	P1,0	P1,0	P1,0	P1,0	P1,0	P1,0	P1,4	P1,4	P1,4
head	P1,3	P1,0	P1,3	P2,3	P2,4	P1,3	P2,1	P1,4	P1,0	P2,2
	P1,2	P1,3	P1,0	P1,3	P2,3	P2,4	P1,3	P2,1	P1,4	P1,0
Stack	P2,2	P1,2	P1,2	P1,0	P1,3	P2,3	P2,4	P1,3	P2,1	P1,4
	P1,1	P2,2	P2,2	P1,2	P1,0	P1,0	P2,3	P2,4	P1,3	P2,1
tail		P1,1	P1,1	P2,2	P1,2	P1,2	P1,0	P2,3	P2,4	P1,3

Faults: 7 Replacements: 6

Exercise S10-1.1

- In a **demand paging** system with **local replacement** policy, **5 frames** are allocated to all **processes**. Pages (and frames) have a size of **16K pages**. The initial state of the system and the page sequence generated by the CPU is detailed below.
- Obtain what pages are allocated on every frame when the next global scope algorithms are applied

a) LRU based on Counters

b) LRU based on stack

c) LRU based in bit reference and 2nd opportunity.

	Inicio	P1,0	P1,3	P2,3	P2,4	P1,3	P2,1	P1,4	P1,0	P2,2
tiempo	20	21	22	23	24	25	26	27	28	29
0	P1,1	P1,1	P1,1	P2,3	P2,3	P2,3	P2,3	P2,3	P2,3	P2,2
1	P2,2	P2,2	P2,2	P2,2	P2,4	P2,4	P2,4	P2,4	P2,4	P2,4
2	P1,2	P1,2	P1,2	P1,2	P1,2	P1,2	P2,1	P2,1	P2,1	P2,1
3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,0	P1,0
4	--	P1,0	P1,0	P1,0	P1,0	P1,0	P1,0	P1,4	P1,4	P1,4
0	1	1	1	1	1	1	1	1	0	1
1	1	1	1	0	1	1	1	1	0	0
2	1	1	1	0	0	0	1	1	0	0
3	1	1	1	0	0	1	1	0	1	1
4	--	1	1	0	0	0	0	1	1	0

Reference bit of page in frame

1 Pointer to the page

Faults: 7 Replacements: 6

Exercise S10-1.1

- Comparative

LRU based on Counters

	Inicio	P1,0	P1,3	P2,3	P2,4	P1,3	P2,1	P1,4	P1,0	P2,2
tiempo	20	21	22	23	24	25	26	27	28	29
0	P1,1	P1,1	P1,1	P2,3	P2,3	P2,3	P2,3	P2,3	P1,0	P1,0
1	P2,2	P2,2	P2,2	P2,2	P2,4	P2,4	P2,4	P2,4	P2,4	P2,2
2	P1,2	P1,2	P1,2	P1,2	P1,2	P1,2	P2,1	P2,1	P2,1	P2,1
3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3
4	--	P1,0	P1,0	P1,0	P1,0	P1,0	P1,0	P1,4	P1,4	P1,4

LRU based on stack

	Inicio	P1,0	P1,3	P2,3	P2,4	P1,3	P2,1	P1,4	P1,0	P2,2
tiempo	20	21	22	23	24	25	26	27	28	29
0	P1,1	P1,1	P1,1	P2,3	P2,3	P2,3	P2,3	P2,3	P1,0	P1,0
1	P2,2	P2,2	P2,2	P2,2	P2,4	P2,4	P2,4	P2,4	P2,4	P2,2
2	P1,2	P1,2	P1,2	P1,2	P1,2	P1,2	P2,1	P2,1	P2,1	P2,1
3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3
4	--	P1,0	P1,0	P1,0	P1,0	P1,0	P1,0	P1,4	P1,4	P1,4

LRU based on reference bit and 2nd opportunity

	Inicio	P1,0	P1,3	P2,3	P2,4	P1,3	P2,1	P1,4	P1,0	P2,2
tiempo	20	21	22	23	24	25	26	27	28	29
0	P1,1	P1,1	P1,1	P2,3	P2,3	P2,3	P2,3	P2,3	P2,3	P2,2
1	P2,2	P2,2	P2,2	P2,2	P2,4	P2,4	P2,4	P2,4	P2,4	P2,4
2	P1,2	P1,2	P1,2	P1,2	P1,2	P1,2	P2,1	P2,1	P2,1	P2,1
3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,3	P1,0	P1,0
4	--	P1,0	P1,0	P1,0	P1,0	P1,0	P1,0	P1,4	P1,4	P1,4

Exercise S10-1.1

- In a **demand paging** system with **local replacement** policy, **4 frames** are allocated to **every process**. The **maximum logical size** of a process is de **4K pages** (3 hexadecimal digits), page size is **64Kbyte**. Suppose that the following table contains all the information related to process P3 at a given time t.

Process P3 information at time t					
Frame (Hexadecimal)	**Page (Hexadecimal)**	**Loading time**	**Last reference time**	**Reference bit**	**Modified bit**
E7	**B72**	**60**	**161**	**1**	**0**
E8	**B71**	**130**	**160**	**1**	**1**
E9	**B70**	**26**	**162**	**0**	**0**
EA	**B73**	**20**	**163**	**1**	**1**

Then process P3 generates the logical address **B745A7C**. Obtain **the physical address** corresponding to this logical address supposing a replacement policy based on 2nd chance replacement algorithm.

Exercise S10-1.1

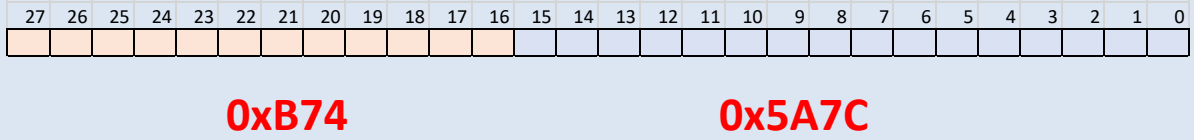
- In a **demand paging** system with **local replacement** policy, **4 frames** are allocated to **every process**. The **maximum logical size** of a process is de **4K pages** (3 hexadecimal digits), page size is **64Kbyte**. Suppose that the following table contains all the information related to process P3 at a given time t=164.

Process P3 information at time t					
Frame (Hexadecimal)	Page (Hexadecimal)	Loading time	Last reference time	Reference bit	Modified bit
E7	B72	60	161	1	0
E8	B71	130	160	1	1
E9	B70	26	162	0	0
EA	B73	20	163	1	1

Then process P3 generates the logical address **B745A7C**. Obtain **the physical address** corresponding to this logical address supposing a replacement policy based on 2nd chance replacement algorithm.

4K pages => 12 bits ; Page size 64K => 16 bits; L. address 12 + 16 = 28 bits.

0xB745A7C => Page: 0xB74 Offset: 0x5A7C



Page B74 => page fault; Assuming that pointer => E7

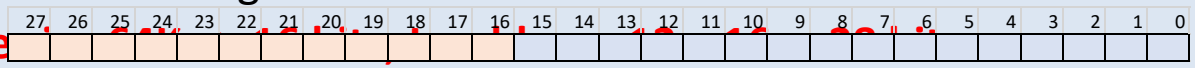
Exercise S10-1.1

- In a **demand paging** system with **local replacement** policy, **4 frames** are allocated to **every process**. The **maximum logical size** of a process is de **4K pages** (3 hexadecimal digits), page size is **64Kbyte**. Suppose that the following table contains all the information related to process P3 at a given time t=164.

Process P3 information at time t					
Frame (Hexadecimal)	Page (Hexadecimal)	Loading time	Last reference time	Reference bit	Modified bit
E7	B72	60	161	1	0
E8	B71	130	160	1	1
E9	B70	26	162	0	0
EA	B73	20	163	1	1

Then process P3 generates the logical address **B745A7C**. Obtain **the physical address** corresponding to this logical address supposing a replacement policy based on 2nd chance replacement algorithm.

4K pages => 12 bits ; Page



0xB745A7C => Page: 0xB74 Offset: 0x5A7C

Page B74 => page fault; Assuming that pointer => E7

Process P3 information at time t					
Frame (Hexadecimal)	Page (Hexadecimal)	Loading time	Last reference time	Reference bit	Modified bit
E7	B72	60	161	0	0
E8	B71	130	160	0	1
E9	B74	164	164	1	0
EA	B73	20	163	1	1

Exercise S10-1.2

- Suppose a **virtual memory system** based on **demand paging**, where **logical addresses are 24 bit wide** and **page size is 1 KByte**. The system can handle up to **1MB of main memory**. The replacement algorithm used is **local 2nd opportunity**
  a) Get the **physical and logical address formats** in this system, indicating the bit number and name of every field
  b) Suppose that at time t = 0 a user requests the execution of process A and the system allocates to it frames 0, 1, 2 and 3. The frames are initially empty and they are allocated in increasing order. Show the **evolution of the physical memory content and how many page faults** would generate the following sequence of logical addresses:

  1000, 3000, 5000, 6000, 7000, 2900, 4900, 900

Exercise S10-1.2

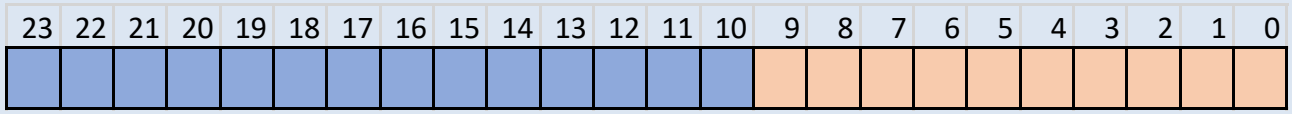
- Suppose a **virtual memory system** based on **demand paging**, where **logical addresses are 24 bit wide** and **page size is 1 KByte**. The system can handle up to **1MB of main memory**. The replacement algorithm used is **local 2nd opportunity**

  a) Get the **physical and logical address formats** in this system, indicating the bit number and name of every field

Logical Address size: 24 bits

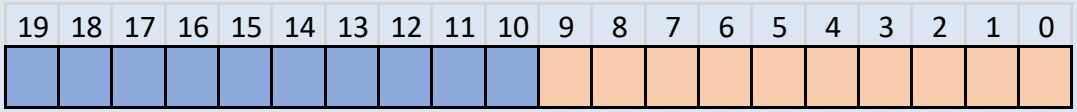
Page size = frame size: 1KB => 10 bits

Logical Address: 14 bits (page) + 10 bits (offset)



Main memory size: 1MB => 20 bits

Physical Address: 10 bits (frame) + 10 bits (offset)



Exercise S10-1.2

- Suppose a **virtual memory system** based on **demand paging**, where **logical addresses are 24 bit wide** and **page size is 1 KByte**. The system can handle up to **1MB of main memory**. The replacement algorithm used is **local 2nd opportunity**

b) Suppose that at time t = 0 a user requests the execution of process A and the system allocates to it frames 0, 1, 2 and 3. The frames are initially empty and they are allocated in increasing order. Show the **evolution of the physical memory content and how many page faults** would generate the following sequence of logical addresses:

Dec	1000	3000	5000	6000	7000	2900	4900	900
Hex	3E8	BB8	1388	1770	1B58	B54	1324	384
Page	**0**	**2**	**4**	**5**	**6**	**2**	**4**	**0**
offset	1000	952	904	880	856	852	804	900

t	0	1	2	3	4	5	6	7
Page	**0**	**2**	**4**	**5**	**6**	**2**	**4**	**0**
0	**0**	0	0	0	**6**	6	6	6
1		**2**	2	2	2	2	2	2
2			**4**	4	4	4	4	4
3				**5**	5	5	5	**0**
0	1	1	1	1	1	1	1	1
1		1	1	1	0	1	1	0
2			1	1	0	0	1	0
3				1	0	0	0	1

- Exercise S10-1: 2nd chance replacement algorithm
  - Exercise S10-1.1: 2nd chance basic
  - Exercise S10-1.2: 2nd chance complete
- **Exercise S10-2: Working set**
  - Exercise S10-2.1: Getting the working set

Exercise S10-2.1

- In a virtual memory system has been decided to use a **working set model** to control physical memory allocation. There are **3 processes A, B y C** in execution. Every memory access is encoded in two characters that represent the process and the page accessed, respectively.

  a) Supposing that the **working set window size (Δ) is 4**, obtain the **working set for every process** at the instant when the last reference happens in the following reference string:

  A0,B2,C3,A0,A1,A5,B2,C4,C2,A2,B1,B3,C0,A1

  C1,B0,A1,C0,B1,B2,C4,A0,B3,B3, C3,A1,C4

  b) Considering that the system has **6 frames**, are they enough to allocate the working sets of all processes till the end?

Exercise S10-2.1

- In a virtual memory system has been decided to use a **working set model** to control physical memory allocation. There are **3 processes A, B y C** in execution. Every memory access is encoded in two characters that represent the process and the page accessed, respectively.
  a) Supposing that the **working set window size (Δ) is 4**, obtain the **working set for every process** at the instant when the last reference happens in the following reference string:

A0,B2,C3,A0,A1,A5,B2,C4,C2,A2,B1,B3,C0,A1

C1,B0,A1,C0,B1,B2,C4,A0,B3,B3, C3,A1,C4

	A0	B2	C3	A0	A1	A5	B2	C4	C2	A2	B1	B3	C0	A1	C1	B0	A1	C0	B1	B2	C4	A0	B3	B3	C3	A1	C4
A	A0			A0	A1	A5				A2				A1			A1					A0				A1	
A						{0,1,5}				{0,1,2,5}				{1,2,5}			{1,2,5}					{0,1,2}				{0, 1}	
B		B2					B2				B1	B3				B0			B1	B2			B3	B3			
B												{1,2,3}				{0,1,2,3}			{0,1,2,3}	{0,1,2,3}			{0,1,2,3}	{1,2,3}			
C			C3					C4	C2				C0		C1			C0			C4				C3		C4
C													{0,2,3,4}		{0,1,2,4}			{0,1,2}			{0,1,4}				{0,1,3,4}		{0,3,4}

Exercise S10-2.1

- In a virtual memory system has been decided to use a **working set model** to control physical memory allocation. There are **3 processes A, B y C** in execution. Every memory access is encoded in two characters that represent the process and the page accessed, respectively.

A	A0			A0	A1	A5				A2				A1			A1					A0				A1	
A						{0,1,5}				{0,1,2,5}				{1,2,5}			{1,2,5}					{0,1,2}				{0, 1}	
B		B2					B2				B1	B3				B0			B1	B2			B3	B3			
B												{1,2,3}				{0,1,2,3}			{0,1,2,3}	{0,1,2,3}			{0,1,2,3}	{1,2,3}			
C			C3					C4	C2				C0		C1			C0			C4				C3		C4
C													{0,2,3,4}		{0,1,2,4}			{0,1,2}			{0,1,4}				{0,1,3,4}		{0,3,4}
						3	3	3	3	**4**	4	4	4	**3**	3	3	**3**	3	3	3	3	**3**	3	3	3	**2**	2
												3	3	3	3	4	4	4	4	4	4	4	4	3	3	3	3
													4	4	4	4	4	3	3	3	3	3	3	3	4	4	3
													11	10	10	11	11	10	10	10	10	10	10	9	10	9	8

b) Considering that the system has **6 frames**, are they enough to allocate the working sets of all processes till the end?

NO. The sum of the Working Set of the 3 processes when the last reference is raised is higher than 6.

The sum of all WSSs is always higher than 6.

Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

Consolidation

Exercises 3

fSO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

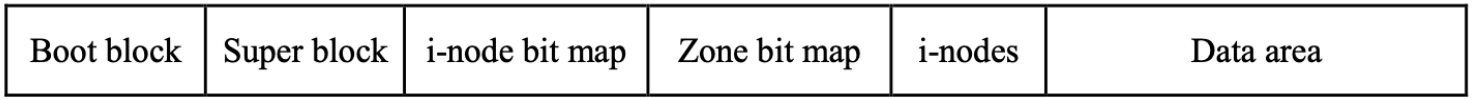
DISCA

Exercise 3.1

A disk partition has a size of 8 Gbytes being formatted with a MINIX file system. The file system has been configured with the following parameters:

- 1 block = 2 KByte
- 1 zone = 1 blocks
- i-node size = 64 bytes with pointers of 32 bits (7 direct, 1 indirect and 1 double indirect)
- 32 bit directory entry
- 16384 i-nodes

The file system structure is :



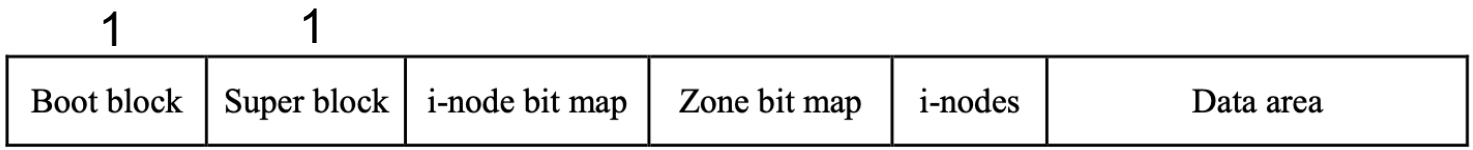
a) Define the sizes of the different file system components

Exercise 3.1

A disk partition has a size of 8 Gbytes being formatted with a MINIX file system. The file system has been configured with the following parameters:

- 1 block = 2 KByte
- 1 zone = 1 blocks
- i-node size = 64 bytes with pointers of 32 bits (7 direct, 1 indirect and 1 double indirect)
- 32 bit directory entry
- 16384 i-nodes

The file system structure is :



a) Define the sizes of the different file system components

i-node bit map: Number of nodes: 16384 16384 bits are needed

In 1 block how many bits are?

1 block => 2 Kbytes => 2 * 1024 * 8 bits = 16384 bits

1 block is sufficient for the i-node bit map

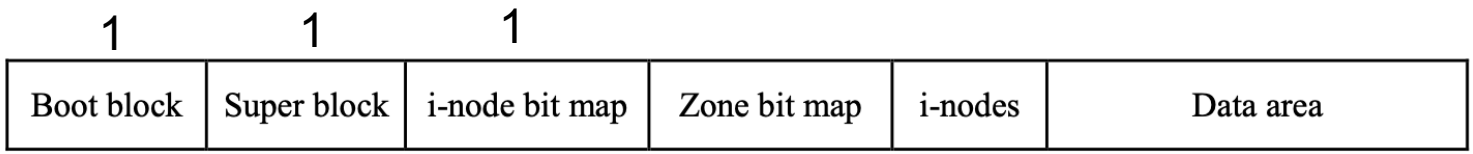
ceiling (16384 / (2 * 1024 * 8)) = 1

Exercise 3.1

A disk partition has a size of 8 Gbytes being formatted with a MINIX file system. The file system has been configured with the following parameters:

- 1 block = 2 KByte
- 1 zone = 1 blocks
- i-node size = 64 bytes with pointers of 32 bits (7 direct, 1 indirect and 1 double indirect)
- 32 bit directory entry
- 16384 i-nodes

The file system structure is :



a) Define the sizes of the different file system components

i-nodes Number of nodes: 16384 16384 * 64 bytes are needed

In 1 block how many i-nodes can be allocated?

1 block => 2 Kbytes => ceiling (2 * 1024 / 64) = 32 i-nodes

So, we need to allocate 16384 i-nodes considering that in each block 32 i-nodes can be allocated

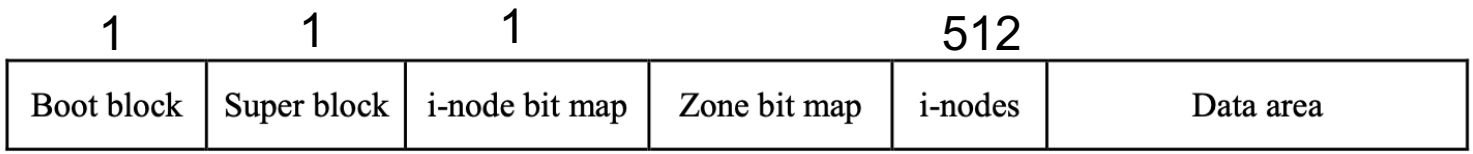
ceiling (16384 / 32) = 512 blocks

Exercise 3.1

A disk partition has a size of 8 Gbytes being formatted with a MINIX file system. The file system has been configured with the following parameters:

- 1 block = 2 KByte
- 1 zone = 1 block
- i-node size = 64 bytes with pointers of 32 bits (7 direct, 1 indirect and 1 double indirect)
- 32 bit directory entry
- 16384 i-nodes

The file system structure is :



a) Define the sizes of the different file system components

How many block are in the file system?
The device has 8GB => ceiling (8GB / 2KB) = 4 * 1024 * 1024 Blocks

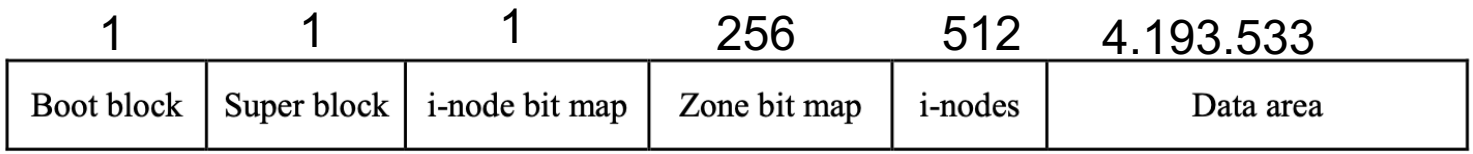
How many zone bitmap blocks are needed?
ceiling (n. zones / bits per area) = ceiling (4 * 1024 * 1024 / 2 * 1024 * 8) = 256 blocks

Exercise 3.1

A disk partition has a size of 8 Gbytes being formatted with a MINIX file system. The file system has been configured with the following parameters:

- 1 block = 2 KByte
- 1 zone = 1 block
- i-node size = 64 bytes with pointers of 32 bits (7 direct, 1 indirect and 1 double indirect)
- 32 bit directory entry
- 16384 i-nodes

The file system structure is :



a) Define the sizes of the different file system components

Total blocks : 4 * 1024 * 1024 Blocks = 4.194.304

Data area: = 4.194.304 – (1 + 1 + 1 +256 + 512) = 4.193.533 blocks

Exercise 3.1

A disk partition has a size of 8 Gbytes being formatted with a MINIX file system. The file system has been configured with the following parameters:

- 1 block = 2 KByte
- 1 zone = 1 block
- i-node size = 64 bytes with pointers of 32 bits (7 direct, 1 indirect and 1 double indirect)
- 32 bit directory entry
- 16384 i-nodes

b) The root directory contains:

- 12 subdirectories (5 empty directories and 2 with 1 regular file of 9KB)
- 5 regular files (size < 1KB)
- 10 regular files (4KB < size < 6KB)
- 2 regular files (size = 20K)
- 1 regular file (size = 1MB)

How many i-nodes are used and how many blocks are occupied.

	i-node	entries	dir areas	pointer block	total areas
root	1	1+1+12+5+10=29	ceiling(29/32) = 1		1
5 empty directories	5	1+1 = 2	1		5
5 reg files (< 1K)	5		1		5
10 reg files [4,6}	10		ceiling(6K /2K) = 3		30
2 reg files [20K}	2		ceiling(20K /2K) =10	1	22
1 reg file (1MB)	1		ceiling(1M/2K) = 512	1	513
2 non empty dirs	2	1+1+1 = 3	1		2
1 regular files	2		ceiling(9K/2K) = 5		10
	28				588

Exercise 3.2

A system has 512 MB of logical addressing space, with 4 KB page size and with 64 MB of physical memory.

a) If memory is managed by paging, with a maximum number of 4 pages per process, the following translation between logical and physical addresses has been done along the execution of process A.

Logical address	10185	28	14050	7248
Physical address	1845193	102428	5858	27728

Fill the page table of process A

Exercise 3.2

A system has 512 MB of logical addressing space, with 4 KB page size and with 64 MB of physical memory.

a) If memoy is managed by paging, with a maximum number of 4 pages per process, the following translation between logical and physical addresses has been done along the execution of process A.

Logical address	10185	28	14050	7248
Physical address	1845193	102428	5858	27728

Fill the page table of process A

Logical address	10185	28	14050	7248
Physical address	1845193	102428	5858	27728

Logical address	27C9	1C	36E2	1C50
Physical address	1C27C9	1901C	16E2	6C50

	2 | 7C9	0 | 01C	3 | 6E2	1 | C50
	1c2 | 7C9	19 | 01C	1 | 6E2	6 | C50

	hex	dec	validity
0	19	25	1
1	6	6	1
2	1C2	450	1
3	1	1	1

Exercise 3.2

A system has 512 MB of logical addressing space, with 4 KB page size and with 64 MB of physical memory.

b) Describe the logical address format if **segmentation with paging** is applied with a maximum of 32 segments per process.

512 MB = 2^{29} B => 29 bits
32 segments = 2^5 segments. => 5 bits
Offset 4KB = 2^{12} Bytes. => 12 bits
Logical address 29 bits = 5 (segment) + [12 (page) +12 (offset)]

Exercise 3.2

A system has 512 MB of logical addressing space, with 4 KB page size and with 64 MB of physical memory.

c) In the previous **segmentation with paging** model, and considering the following page table of segment 1 for a given process, obtain the logical addresses that correspond to physical addresses 8220 and 25096 in that process.

Segment 1		
page	frame	Valid
0	4	1
1	5	1
2	6	1
3	2	1

8220	25096
201C	6208

8220 => 0x201C => 2 | 01C. Frame 2 allocates page 3 of segment 1

Logical address S1, 3 | 01C = S1, 3 * 4096 + 28 = S1, 12316

25096 => 0x6208 => 6 | 208. Frame 6 allocates page 2 of segment 1

Logical address S1, 2 | 208 = S1, 2 * 4096 + 520 = S1, 8712