

# OS Memory

Sept 17, 2018

## Release\_mem()

- If release\_mem() is called but there is no memory to release, is that an error?

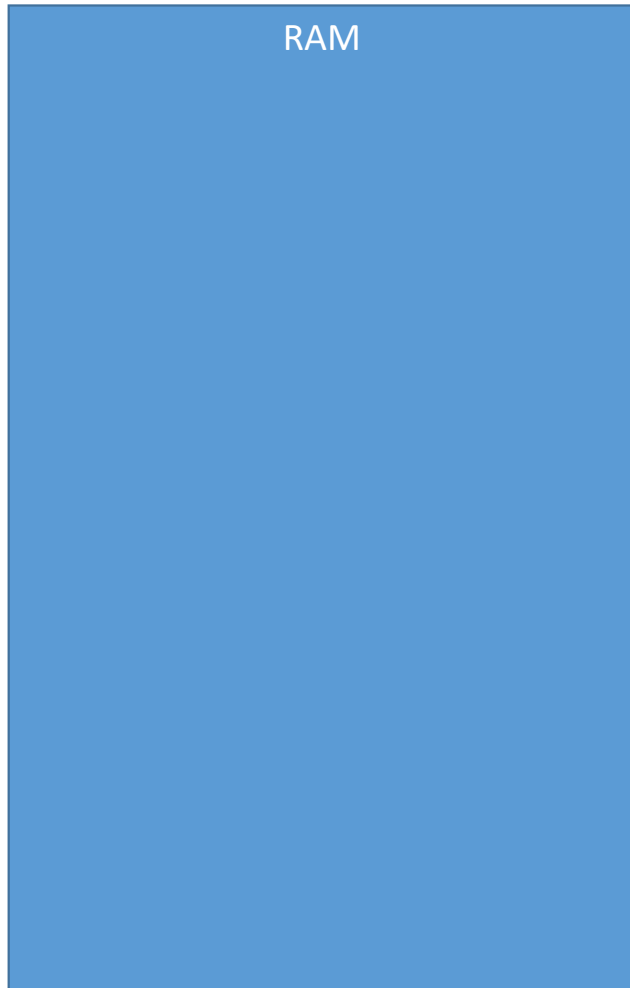
**NO, but we should log this as this is a serious error  
BUT we should not stop the user working**

# Observe behavior of request and memory availability

- Assumption: contiguous memory
- Requests ask for 3 blocks, 2 blocks and 1 block
- SUGGESTION: Why don't we make partitions where partition is a certain number of block memory
  - So, one partition consists of 3 blocks, another partition of 2 blocks, etc
  - Would this be better??

**Finish off the contiguous memory scheme**

## Scheme-2 (Fixed Partition)



RAM

We call each memory area a Partition.  
Each partition has a fixed size.  
We need to keep track of memory used.

JOB 1

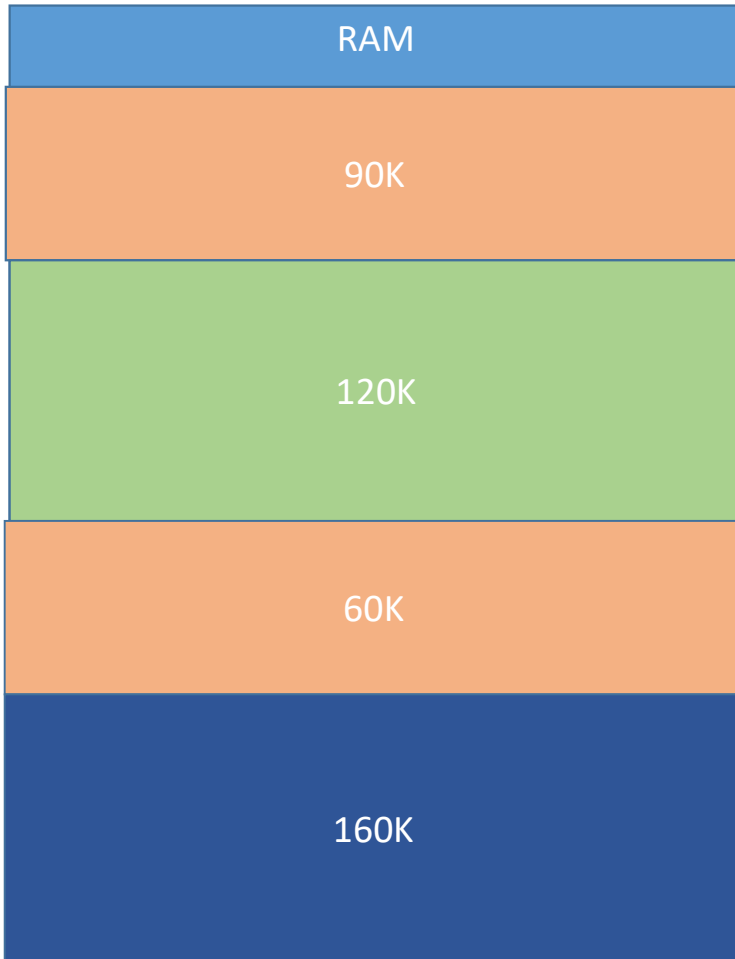
PARTITION MEMORY TABLE

Size	Start Address	Name of Job	Status
60K	100K	JOB 1	BUSY
200K	180K	JOB 2	BUSY
100K	280K		FREE
50K	380K	JOB 3	BUSY

Size of partition is static. Meaning the size cannot be changed.

JOB 3

Only way to change the size is to reboot the computer.



The sizes of partitions are fixed at system start.

## Issues with scheme-2

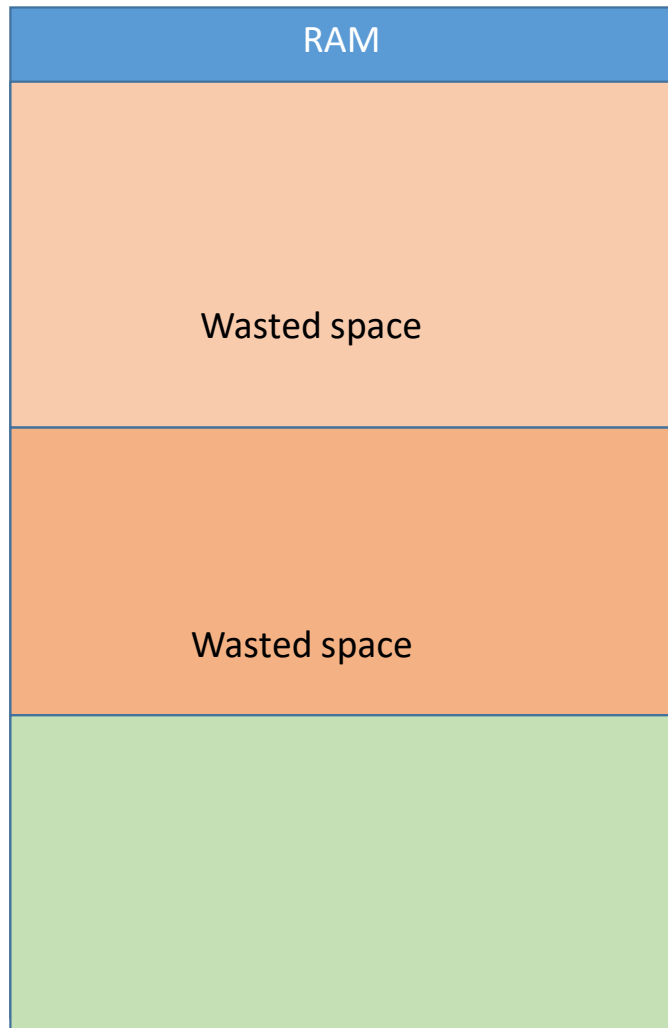
- Memory sizes are fixed, and only way to change them is to either reboot the system, or clear memory and start again
- Some jobs may not be able to find the right memory partition because the memory partition sizes are fixed.
- To change the size we have to reboot.
- We need a scheme where memory sizes can be dynamic upon on request or demand.

## Scheme-3: Dynamic Partition

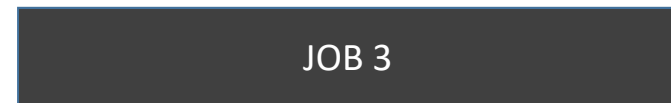
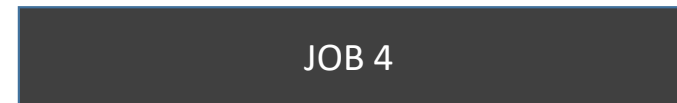
Your code is based on this model.  
Use the idea of a block of memory.  
Process **requests** number of blocks.

- Don't predetermined the size of the memory partition
- Only allocate the memory on request
  - When a job requests 100K of memory
  - Allocate 100K of contiguous memory to job
- This solves the problem with fixed partition on having to reboot to adjust the size of memory partition
- It does prevent wasted memory

**Windows and Linux allocate memory  
Request in chunks of 256K, 512K or 1024K**



## Two ways to fit job into memory for fixed and dynamic partition



**First fit all**

Lots of wasted space

**First fit Allocation**

Find the smallest partition to fit

Slower performance in finding memory  
Better efficiency in use Of memory



**When a job is finished, it is removed from memory**

**This is called DeAllocation**

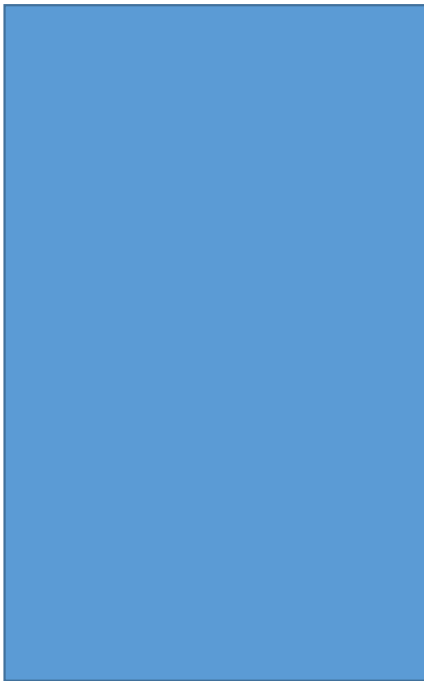
**For fixed partition, the job is just removed and the size of partition remains the same.**

**For dynamic partition, the job is removed and the manager will try to join two adjacent memory partitions into one big partition.**

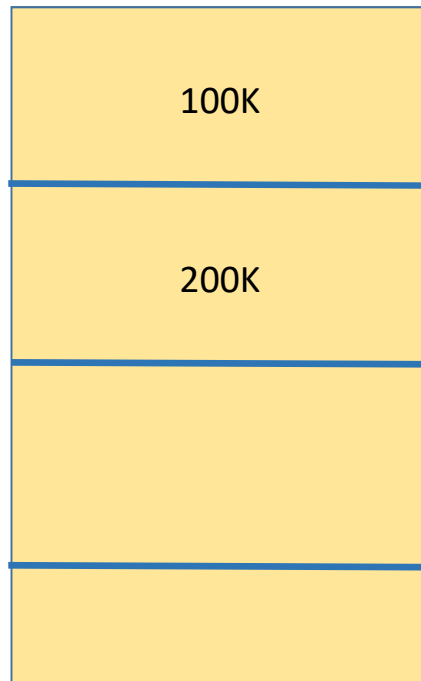
# Issues with these 3 schemes

- Each scheme requires that the whole program be stored in memory in a single contiguous block within a partition.
- Can we divide the program into multiple smaller chunks?
  - YES, we can.
  - This brings us to the topic of Virtual Memory

Scheme-1 = the whole memory. One job at a time

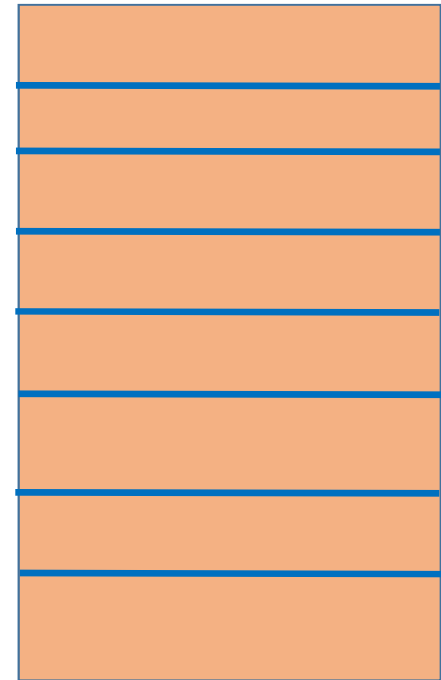


Scheme-2. Fixed partition of various sizes. Allows multiple programs to run.



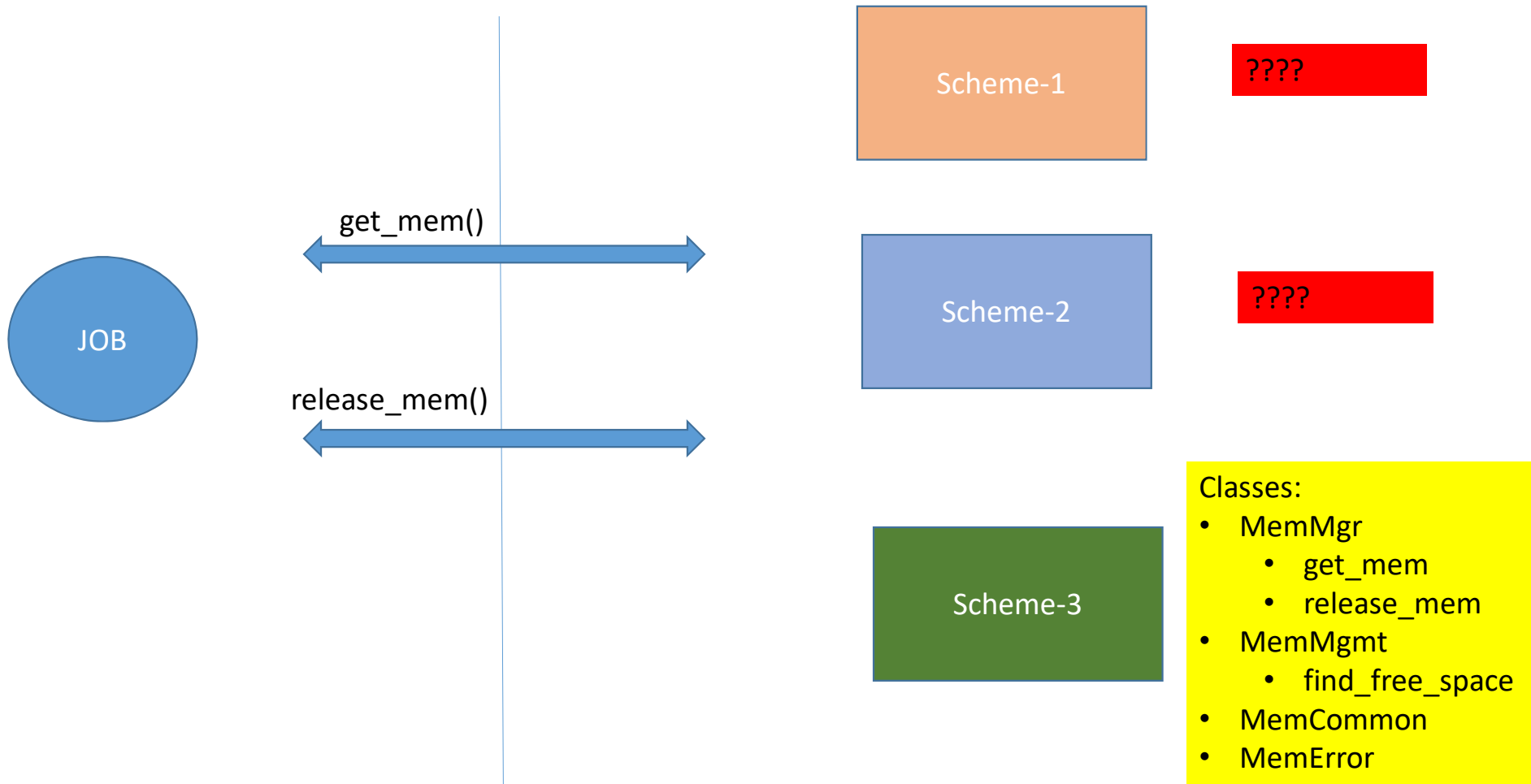
Your code implements this scheme.

Scheme-3. Fixed partition of same size. Allows multiple programs to run.



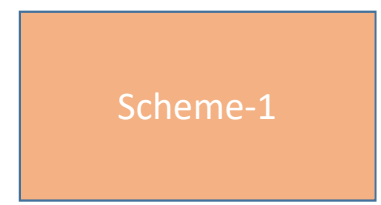
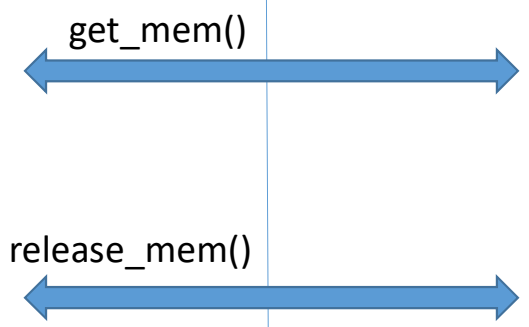
# The test

- Your code is the implementation of scheme-3 with two API (Application Programming Interface) to the “programs”.
  - `get_mem(pid, numberOfBlocksOfMemory)`
  - `release_mem(pid)`
- What changes are required to convert your code to support scheme-1 and scheme-2?

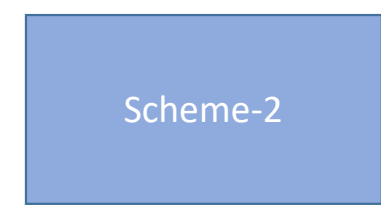


# OS Memory

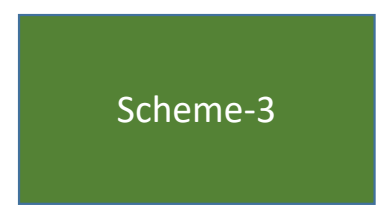
Sept 19, 2018



????



????



- Classes:
- MemMgr
    - get\_mem
    - release\_mem
  - MemMgmt
    - find\_free\_space
  - MemCommon
  - MemError

Testmem2.py  
Has access to MemMgr and MemError and no  
Other classes.  
The classes MemCommon, MemMgmt and others are not  
Visible to users.

Who owns these data structures?

Memory Array

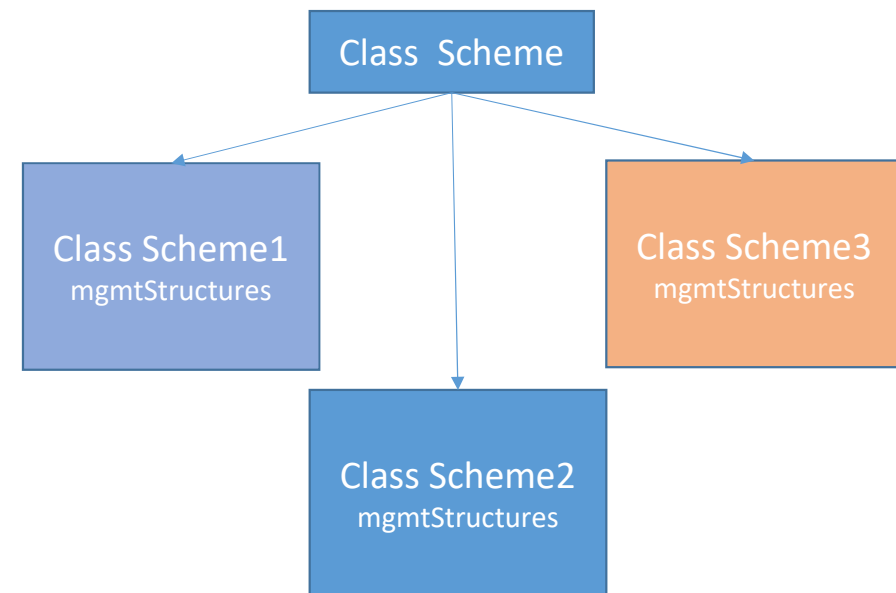
Management Structures

Class MemX  
memoryArray  
managementStructures

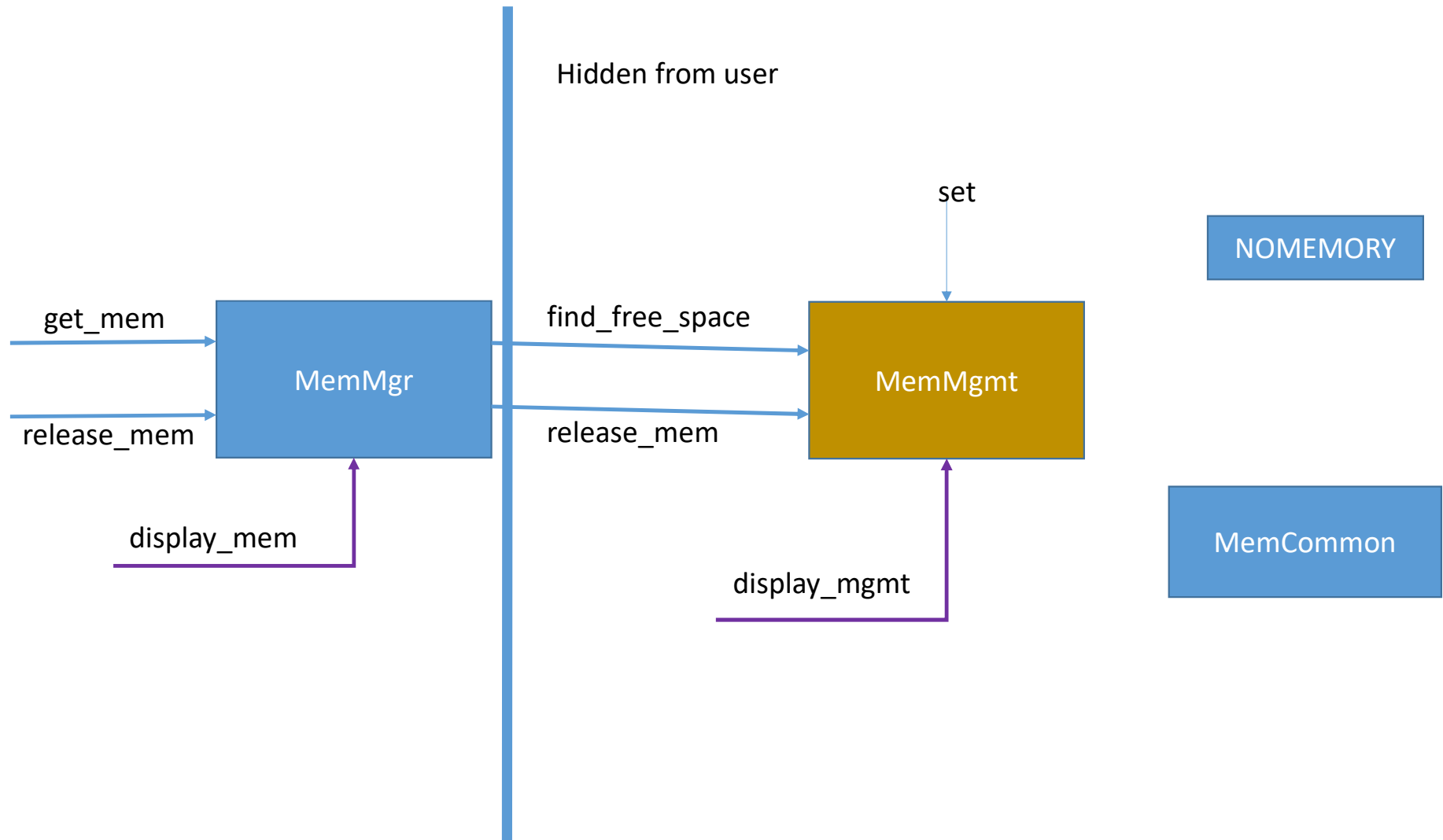
Class MemZ:  
memoryStructures

Class MemY  
memoryArray

Class MemCommon:  
memoryArray  
managementStructures







```
class MemMgr:

    memarray

    @classmethod
    def get_mem(...):
        mgmt.find_free_space(..)
        return array

    @classmethod
    def release_mem(..)
        mgmt.release_mem()

# debugging methods

    @classmethod
    def print_mem()
```

Class MemMgmt:

```
    mgmtarray

    @classmethod
    def find_free_space(..)
        set(...)
        raise NOMEMORY
        return start_index, end_index

    @classmethod
    def set(pid, start_index, end_index)

    @classmethod
    def release_mem(pid)

# debugging method

    @classmethod
    def print_mgmt():
```

```
class MemCommon:
```

```
# memory array
```

```
    mem_row_size = 7
```

```
    mem_column_size = 6
```

```
# management array
```

```
    mgmt_row_size = 7
```

```
    mgmt_column_size = 1
```

```
    mgmt_pid_column = 0
```

```
    mgmt_free = 0
```

```
class NOMEMORY:  
    pass
```

### **RECOMMENDATION:**

**Bring your code up to date.**

**Get a copy from your friends who got it right.**

**We need the correct code for the next stage**

**To answer the questions, you need to  
Know the memory array and  
management structures**

Scheme3

Index	PID
0	222
1	0
2	0
3	0
4	0

Assumptions:

Memory is divided into  
Fixed row size and  
Fixed number of rows.

Users ask for number of blocks  
Users get a subarray

Scheme1

Index	PID
0	222

Assumptions:

Memory is divided into  
One big row of known  
Size

Users ask for a number of blocks.  
Always get the whole memory

Scheme3

Partition	First Index	PID	Size(nbr of blocks)
1	0	222	3
2	3	333	1
3	4	0	3
4	7	0	3
5	10	0	2
6	12	555	5

Assumptions:

Memory is divided into multiple  
Partitions of different sizes (number of row)  
Users ask for a number of blocks.  
Return a subarray

find\_free\_space will do a first\_fit and  
best\_fit

# Some questions

**Provide me a direct answer.**

- What does get\_mem return?
  - Returns an array – For scheme1 – the whole array, For scheme-2, it is a subset of view of the partition
  - get\_mem is a function.
- Changes to get\_mem
  - Talk about what get\_mem internally need to do
- Changes in management structures
  - Talk about changes and why these changes are required

# Use of language

- Get to the point
- Talk about process and what's need to be done
- Provide context
- Don't assume

# Virtual Memory

Providing a way to fit a big program into memory

# Written Assignment: written-sept19-a

## DUE: Sept-24

- Read OS Memory Management paper provided in folder “reference”
- Explain how you would implement paging based on the current code base
  - Use diagrams
  - Use precise language – or pseudo code



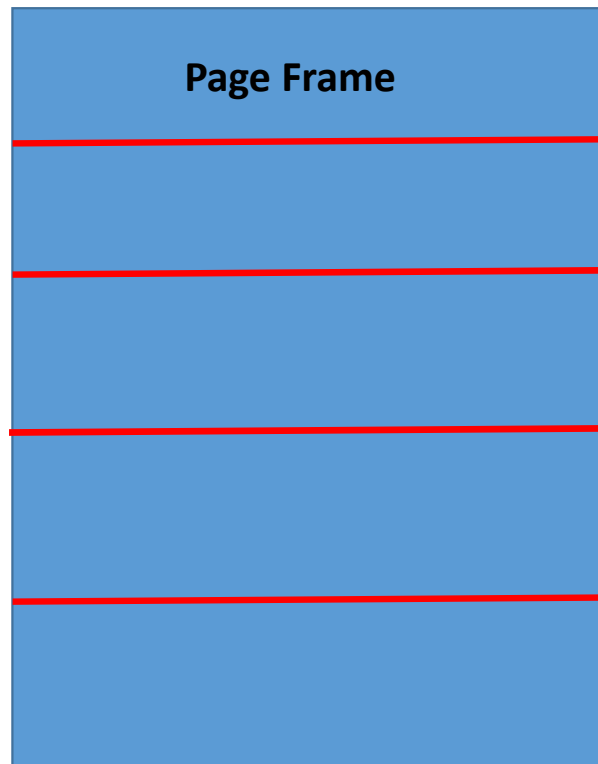
# Paged memory Allocation

- Divide the job or program into equal size chunks
  - The sections within a disk sector are called Blocks
  - The sections within memory are called Page Frames.
- 
- When a disk is formatted, the disk is divided into sectors and blocks
  - On computer startup, the memory is partitioned into page frames
  - When the program is divided into blocks we refer to each block as a page
  - The container for each program page is called the page frame

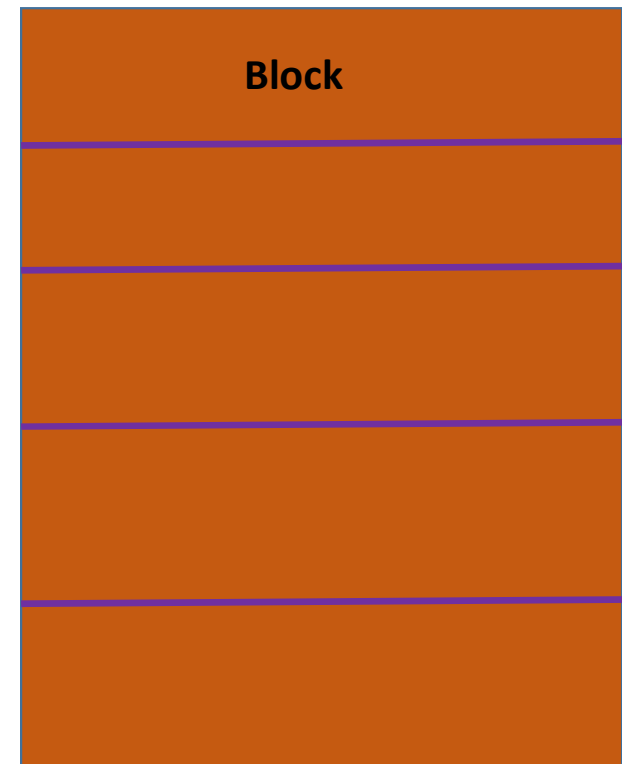
## Program



## Memory



## Disk



# Segments and Pages

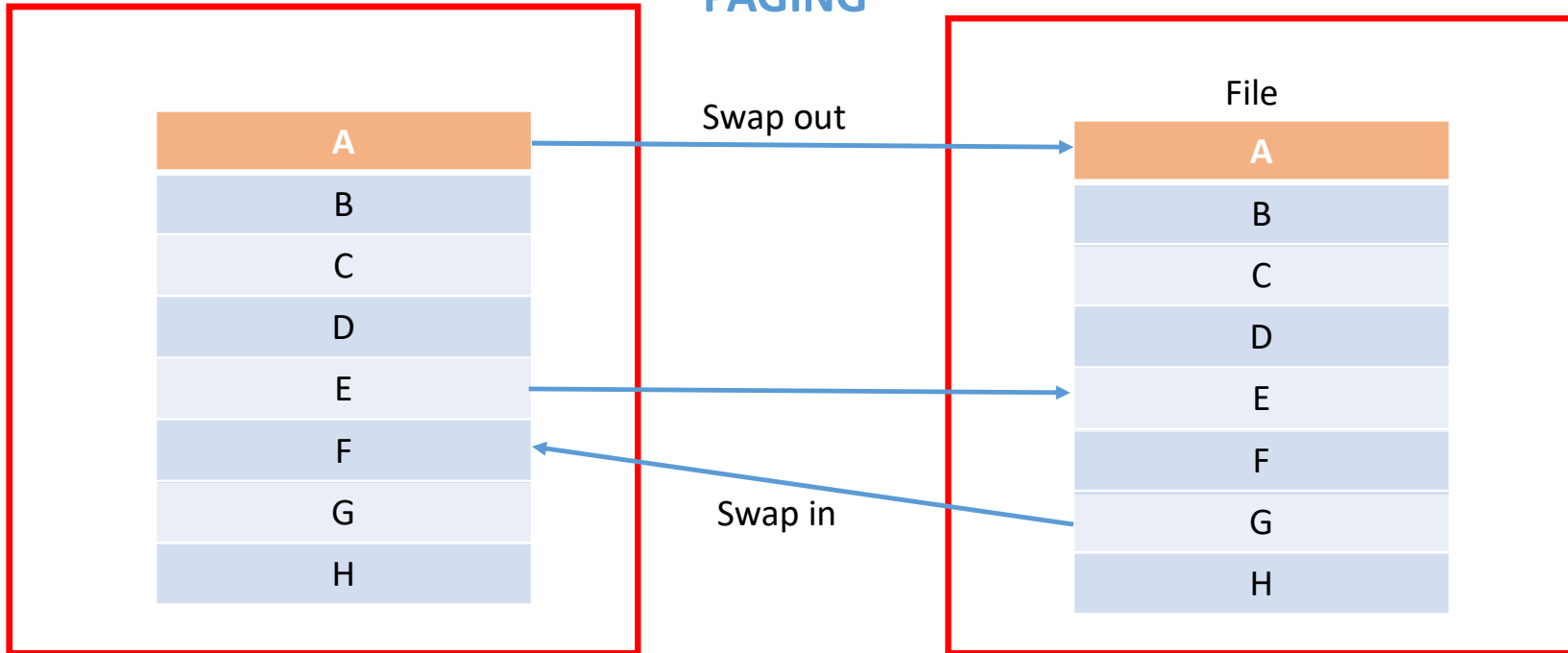
- A Page is a fixed size block – similar to scheme3
- A Segment is a variable size – similar to scheme2
- Hardware decides on the size of a page
- User decides on the size of a segment
- Paging leads to internal fragmentation
- Segmentation leads to external fragmentation

Concerning your code assignment

## MEMORY

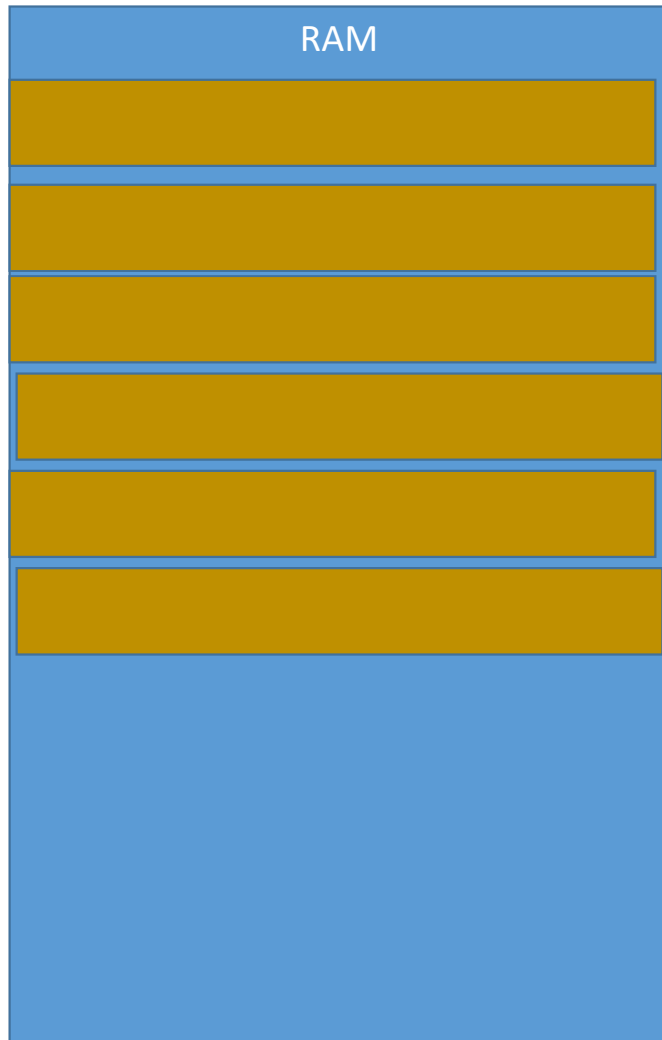
## PAGING

## SECONDARY STORAGE



Page A must go to page A in file independently.

Page F from file could be loaded into page G in memory.



**When a program makes a request for memory,  
The OS MM makes a determination on how many pages  
are in the program.**

**REMEMBER, page frames are fixed size. Each page frame is the  
same size.**

**The OS MM finds empty page frames to fit  
all the pages of the program.**

**The page frames may not be adjacent to  
one another.**



**We need data structures to keep track of all these  
Page frames and programs or jobs in memory**

# Virtual Memory

Sept 21, 2018

# Reasons why the entire program need not be in memory

- Error routines are only used only for errors. Code is therefore not used until required
- Certain functions are only used in rare occasions
- Some data manipulations are only used in some small part of data structures like a table

**Is this what happens to you?**

**You see the requirements. You see diagrams.**

**You roll your eyes.**

**Your life flashes before your eyes**

**How to start? Where to start?**

**You wonder whether it's still too late to switch major to ... say social studies.  
At least there are humans... and not machines**



# The secret art of software development

**Test driven method  
and incremental development**

**Test driven – write the test program first  
– from a simple one to more complex**

**Incremental Development – develop the software bit by bit**

**Start with the easiest (usually). Follow the flow of your test.**

**Software  
Engineering**

Everything (most of it) will be relevant to the paper you are writing

- Due date of paper: Sept 24<sup>th</sup>
- Goal of paper:
  - How would you implement paging?
  - Show with diagrams and sketching out some flows, and maybe even classes
  - Put down assumptions and explain
  - No coding..yet

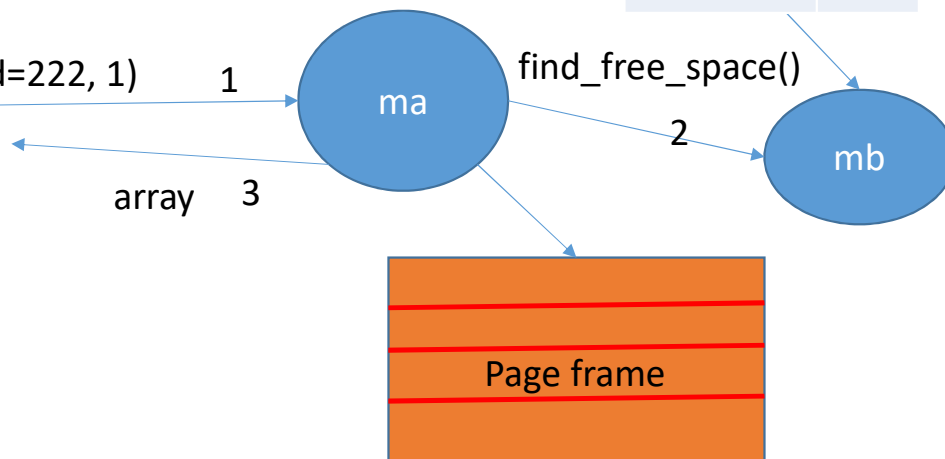
Ask me questions?

Logical	Physical		
A	0		
B	-1		
C	-1		

Process Memory Mgmt

Test Program

ma.get\_mem(pid=222, 1)



Page Frame	PID
0	222
1	0

Physical Memory Mgmt

Physical Memory

Suppose we have this:

```
ma.get_mem(222,1)
ma.get_mem(333,1)
ma.get_mem(222,1)
```

Page Frame	PID
0	222
1	333
2	222

**How will the process store the next call to get\_mem()**

**Do it yourself for the first 5 minutes**

**THEN get into your assigned groups and discuss**

Logical	Physical		
A	0		
B	-1		
C	-1		

How does the process know how many pages are in the program?

Before `get_mem()` is called,  
The loader will query the program file

- What is your size
- How many pages
- Minimum memory to execute
- Request for a process ID
- Indicate when the process will start
- Initialize the process
  - Construct its own memory mgmt
  - `get_mem()`
  - Request pages from file
    - Call file mgr to load those pages

Logical	Physical
A	
B	
C	
D	
E	
F	

DISK

A
B
C
D
E
F

# Steps to a software design

- Write out your assumptions
- Draw out the data and data structures required – global or local
- Start with the interface call and follow it through those data structures and data, until it returns something to the caller
- If there are missing information along the trace, fill it out and start over again
- Consider multiple scenarios

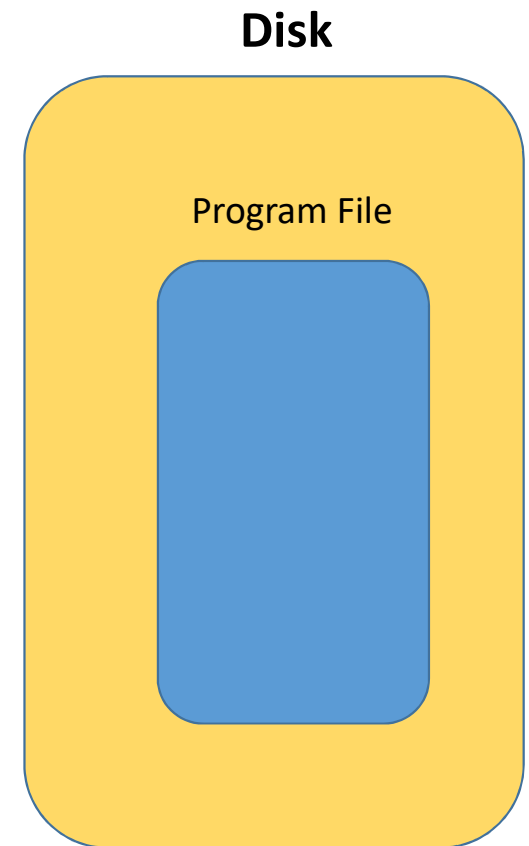
# Groups

A	B	C
Aaron	Daniel	Quentin
Devon	Dylan	Tristan
Elyssa	Tyler	Gabriel

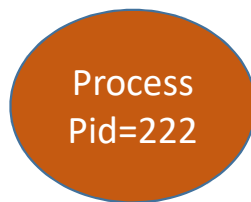
Each group takes one third of the board

### ASSUMPTIONS:

- Memory is partitioned into fixed size
- Partition is called a Page Frame
- Program File is divided into fixed size called Pages
- Page and Page Frame are of equal size
- Memory Mgmt manages Page Frames where each page frame is indexed from 0 to n.



Memory Mgmt



Management Structures




What is the first call?  
What is the 2<sup>nd</sup> call and others?



MEMORY MAP TABLE	
PAGE FRAME	STATE
3	FREE
4	FREE
5	BUSY

PAGE MAP TABLE (PMT)	
PAGE NUMBER	PAGE FRAME NUMBER
0	5
2	4
1	3



**Each job or program will get its own Page Map Table (PMT)**

JOB TABLE		
JOB NAME	JOB SIZE	PMT LOCATION
JOB 2	400	3096
JOB 5	200	3100
JOB 44	800	3150

Address location