Operating Systems, Security and Networks (207SE)
Lab 6: **Memory management and virtual memory**

In this tutorial we will explore approaches used by operating systems in manage the memory requirement of the processes.

<u>Setting up the activities</u>

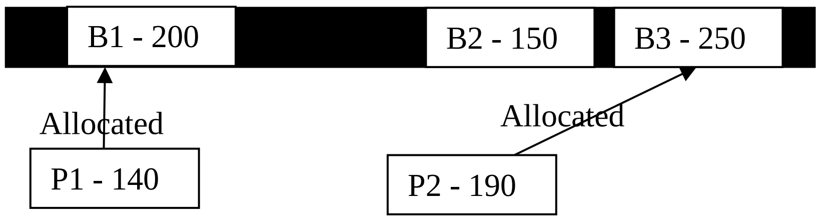
Getting to required exe file

- Run putty and connect to server
- Use wget http://creative.coventry.ac.uk/~james/c-code.zip
- unzip c-code.zip
- cd c-code
- chmod 555 *

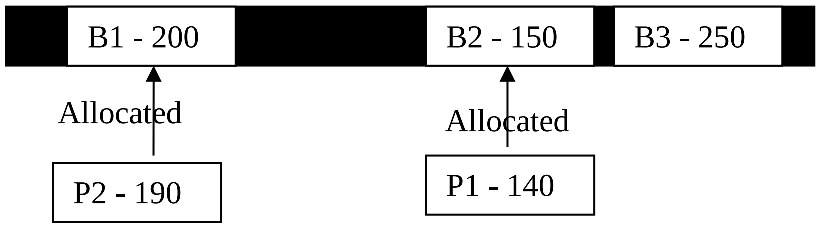
<u>Memory Allocation</u>

When a process is created, it requires memory to be allocated to it. In this exercise you will look at three approaches that operating systems use to allocate the memory required by the processes to the available blocks in main memory. The approaches we will look at here are first-fit, best-fit and worst-fit. **Note:** these are simplified version of the ones actually used.

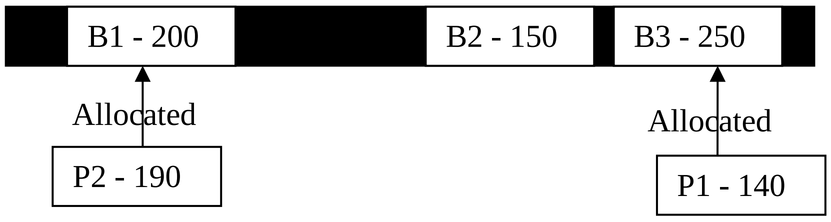
First-fit approach - the process is allocated to the first unallocated memory block in which it fits.



Best-fit approach - the process is allocated to a block of unallocated memory in which it fits with the smallest memory left over.



Worst-fit approach - the process is placed in the largest block of unallocated memory available.



Portfolio Exercises

1. For the example below determine using the first-fit memory allocation method which unallocated memory blocks the process memory requirements will be allocated to.

   5 memory blocks available
   5 processes require memory

   Memory unallocated in block 1: 300
   Memory unallocated in block 2: 500
   Memory unallocated in block 3: 250
   Memory unallocated in block 4: 220
   Memory unallocated in block 5: 270

   Process 1 requires memory size of: 300
   Process 2 requires memory size of: 350
   Process 3 requires memory size of: 450
   Process 4 requires memory size of: 400
   Process 5 requires memory size of: 150

M1 (300)	M2 (500)	M3 (250)	M4 (220)	M5 (270)

   Run the exe file first-fit using ./first-fit and see if you got the same result.

   Note: When asked the number of memory blocks enter 5 and the same for processes

2. Repeat the same activity as above but for the best-fit and worst-fit methods. Once you have done the allocation approach yourself use the exe code by typing ./best-fit then ./worst-fit.

3. Which approaches allocates all of the processes and with the least fragmentation.

   Best fit

M1 (300)	M2 (500)	M3 (250)	M4 (220)	M5 (270)

   Worst fit

M1 (300)	M2 (500)	M3 (250)	M4 (220)	M5 (270)

Virtual memory paging approaches

Paging is the process by which the operating system makes it appear that the memory available is not limited by the size of the main memory. It achieves this by moving the data and execution components required by the process in and out of main memory. It keeps track of this by using a page table containing what pages are in main memory and their physical addresses. There is a limit to the number of pages that can be in main memory at any time and so the number of enters in the page table. If the process needs a page that is not in main memory, the operating system must move a page in from disk, if the available space in main memory is full swap a page out of memory and then update the page table. When there is a need to move a page from backup store this is known as a page fault; if the page is in main memory and found in the page table this is known as a touch.

A page table in a real computer system will have over one thousand enters, but to keep it simple in our portfolio activity we will limit the available main memory to 3 or 4 pages. In this portfolio you will be provided with the order that the pages are required by the process. Of course in real-time computer systems the operating system will not know this. The two methods covered in this portfolio activity are:

The first in first out method: If there is a need to swap out a page the one that has been in memory the longest is moved out.

The random method: The page selected to move out of memory is done using a random.

Portfolio Activities

1. Using the first-in-first-out paging approach, complete the paging table below showing how pages will be swapped in and out of main memory for a paging table with limits of 3 or 4 table entries. Determine the page fault total.

   Paging Accessing Sequence: 42775639322

   Three table page entries

	4	2	7	7	5	6	3	9	3	2	2
Page Entry 0											
Page Entry 1											
Page Entry 2											
Page Fault											

   Page Fault Total:

Four table page entries

	4	2	7	7	5	6	3	9	3	2	2
Page Entry 0											
Page Entry 1											
Page Entry 2											
Page Entry 3											
Page Fault											

Page Fault Total:

2. Run the fifo exe file (./fifo) is the result the same

3. Repeat the above process for the random page allocation approach.

   Paging Accessing sequence: 42775639322

   Three table page entries

	4	2	7	7	5	6	3	9	3	2	2
Page Entry 0											
Page Entry 1											
Page Entry 2											
Page Fault											

   Page Faults Total:

   Four page entries

	4	2	7	7	5	6	3	9	3	2	2
Page Entry 0											
Page Entry 1											
Page Entry 2											
Page Entry 3											
Page Fault											

   Page Fault Total:

4. Run the random exe file (./random) is the result the same. If not why not