

MONASH UNIVERSITY
INFORMATION TECHNOLOGY
Clayton School



FIT2022 Computer Systems II

AJH

MUSO | About FIT2022 | Assessment | Contacts | Laboratories | Lectures | Resources | Timetables | Tutorials

Last modified: 20070926:145440/added marking scheme

FIT2022 AJH-2007-18

Assignment 2

[Objectives](#) | [The Banker's algorithm](#) | [Page Replacement](#) | [Any Questions?](#) | [Submitting your Work](#)

Due Date: 3 Oct 2007

1. Objectives

The objectives of this assignment are to:

1. develop skills in applying the Banker's algorithm; and
2. model a range of page replacement algorithms

A [pdf version](#) of this assignment sheet is also available.

2. The Banker's algorithm

Consider the following system snapshot, with resources A, B, C and D and processes P₀ to P₄:

	Max				Allocation				Need				Available			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P ₀	0	0	1	2	0	0	1	2								
P ₁	2	7	5	0	2	0	0	0								
P ₂	6	6	5	6	0	0	3	4								
P ₃	4	3	5	6	2	3	5	4								
P ₄	0	6	5	2	0	3	3	2								
													2	1	0	0

Using the Banker's algorithm, answer the following questions:

1. How many resources of type A, B, C and D are there?
2. Compute what resources each process might still request, and display them in the Need matrix.
3. Is the system in a safe state? Why?
4. Is the system currently deadlocked? Why?
5. If a request from P₂ arrives for additional resources of P₂(0, 1, 0, 0), can the algorithm grant the request immediately? What state would the system be if the request is granted immediately? Which processes, if any, are or may become deadlocked if this whole request is granted immediately?

Write your solutions into a pdf file called 'banker.pdf'.

3. Page Replacement

You are to write a Python program to model a range of page replacement algorithms. The program should be structured in such a way that each algorithm can be dynamically loaded and executed, to see the effect of using different algorithms on the same page reference string.

3.1 The Page Replacement Algorithms

3.1.1 The parent class PRA

Each page replacement algorithm shares a number of common methods. You are to write a superclass, `PRA`, contained in a file `PRA.py`, that has the following variables/methods (besides the usual initialization, etc.):

- faults**
the number of page faults that have occurred.
- pageouts**
the number of pages that have had to be written out
- ref(p,w)**
a reference to page number `p` is made. Returns the frame number `f` at which the page resides. If a page fault occurs, increment `faults`, and invoke the `replace` method. 'w' is the write bit, indicating that the page has been written since the copy on disk.
- replace(p)**
Find a frame `f` to remove, and replace it with `p`. If the frame `f` is dirty, increment the `pageouts` variable. Return `f`.

Each of the following algorithms must inherit from `PRA`.

3.1.2 First-In First-Out

The frame selected for removal is the oldest page amongst all those in main memory. Write a python class `fifo` (in a file called `fifo.py`) that inherits from `PRA` and implements first-in first-out.

3.1.3 Least Recently Used

The frame selected for removal is that frame that has not been referenced for the longest period. Write a python class `lru` (in a file called `lru.py`) that inherits from `PRA` and implements least recently used.

3.1.4 Least Intensively Used

The frame selected for removal is that frame that has had the fewest references (since being loaded) of all frames in memory. Use FIFO to break any ties. Write a python class `liu` (in a file called `liu.py`) that inherits from `PRA` and implements least intensively used.

3.2 The Main Program

Your main program is responsible for implementing all the tests required under section "Testing your Page Replacement Program" below. Arrange your main program to test both the given reference string, followed by at least two reference strings (with the same parameter) generated by the page reference generator, given in the next section.

A key aspect of your main program is that it must dynamically load the code for each of the three replacement algorithms. That is, they should not be statically included in the code for the main program, but rather, read in at run-time as a string (from the various files), and then `eval`d to define the appropriate class. The name of the file should be used to determine the class to instantiate.

The algorithms to test must be defined by a list, which (as an extension exercise) can be augmented to add other replacement algorithms if desired.

Your main program should therefore contain the following fragments (or similar - feel free to use `eval/exec/execfile` as appropriate, or to structure the body of the for loop in whatever way you wish - the only essential thing is that the replacement methods must be called dynamically, as per the `# invoke the ref method dynamically` line:

"paging.py" 3.1 =

```

algs=['fifo', 'lru', 'liu']
refstring=[3,4,6,8,7,5,1,3,1,5,6,0,4,5,2,7,4,3,2,7,0,3,3,6,3]
framesizes=[3,4,5]
alpggm={}
...
for alg in algs:
    execfile(alg+'.py') # define the replacement class
    alpggm[alg]=eval(alg) # instantiate the class
    ...
    for fsize in framesizes:
        ...
        for p in refstring:
            ...alpggm[alg].ref(p,w) # invoke the ref method dynamically
        ...
    ...

```

3.3 Page Reference Generator

The following code is provided to assist you in your testing.

"pagerefgen.py" 3.2 =

```

#!/usr/bin/python

import getopt
import random
import sys

pages=12
length=12

def usage():
    print """

        pagerefgen.py [-f npages | --pages=npages] [-l nrefs | --length=nrefs]

This program generates a random sequence of page numbers. The page
numbers are drawn from a logical address space containing npages
pages. The sequence will be of length nrefs.

    npages = the number of pages in the virtual address space
    nrefs  = the total number of page references in the address string

    """

def generate(n,l):
    seq=[]
    for count in range(l):
        nextref=random.randint(0,n)
        seq.append(nextref)
    return seq

if __name__ == '__main__':
    (vals,path)=getopt.getopt(sys.argv[1:], 'p:l:',
                              ['pages=', 'length='])
    for (opt,val) in vals:
        if opt=='-p' or opt=='--pages':
            pages=int(val)
        if opt=='-l' or opt=='--length':
            length=int(val)

```

```

if len(path)!=0:
    usage()
    sys.exit(1)
seq=map(str,generate(pages,length))
for s in seq:
    print "%s" % s,

```

This program can either be used stand-alone to generate a string of page references, or imported into another python program to generate a list of page references. In the latter case, the usage is through calling the function `generate(n,l)`, where `n` is the number of pages in the logical address space, and `l` is the length of the generated reference list.

3.4 Testing your Page Replacement Program

You should run your `paging.py` program with at least the following reference string on your three algorithms:

3 4 6 8 7 5 1 3 1 5 6 0 4 5 2 7 4 3 2 7 0 3 3 6 3

You should test your algorithms for memory sizes of 3,4,5 frames, and show the results in a pdf file `replacement.pdf`. Comment on the differences in results in your discussion.

4. Any Questions?

If there is anything unclear, or something you think is needed is omitted, ask your question in the [Anonymous Feedback Page](#) and it will be answered as soon as possible.

5. Submitting your Work

MOST IMPORTANT! Your work is to be submitted through MUSO as a tar file (uncompressed). The tar file should be named `12345678.tar`, where 12345678 is your student ID number. It should contain

1. `banker.pdf` the answers to the Banker's algorithm of section 2.
2. `paging.py` the complete Python text of your program, in runnable form
3. `PRA.py` the PRA class.
4. `fifo.py` the First-in, First-Out replacement algorithm.
5. `lru.py` the Least Recently Used replacement algorithm.
6. `liu.py` the Least Intensively Used replacement algorithm.
7. `replacement.pdf` a pdf file containing your answer to the page replacement problems.

Do not submit these files directly as they will be discarded! You must also take care to use the names exactly as given (observe case of letters).

and please, NO Word, rar, or zip files!!

Your work will be marked according to the following schema, which will be returned as an attachment under MUSO:

"[marks.txt](#)" 5.1 =

```

Assignment 2

Your name, Your student ID number
PLEASE MAKE SURE YOU INCLUDE THESE ON ALL FILES!

Q2 The Banker's Algorithm
  2.1 /1
  2.2 /1
  2.3 /1
  2.4 /1
  2.5 /2

Q3 Page Replacement

```

3.1 PRA	/2
3.1.1 FIFO	/2
3.1.2 LRU	/2
3.1.3 LIU	/2
3.2 Main (replacement.pdf)	/4
3.4 Test	/2

Document History

20070926:145440 1.0.3 ajh added marking scheme
20070926:122613 1.0.2 ajh added clarification about PRA class and file
20070829:143426 1.0.1 ajh added dynamic loading details
20070828:121402 1.0.0 ajh first version released - not complete

This page maintained by John Hurst.

Copyright [Monash University Copyright Policy](#)

Generated at 20070926:1458 from an XML file modified on 20070926:1458

Maintainer use only; not generally accessible: [Local Server](#) [Work Server](#) [CSSE Server](#)

40 accesses since
18 Jul 2007

