Mittuniversitetet
MID SWEDEN UNIVERSITY

Laboratory assignment: Paging Algorithms

Daniel Bosk*

paging.tex 187 2016-11-27 23:09:21Z jimahl

Contents

1 Introduction **1**

2 Aim **1**

3 Reading Instructions **2**

4 Theory **2**
4.1 The Simulator Written in C 2
4.2 The Simulator Written in Python 3

5 Work **5**

6 Examination **5**

A A Report Template **6**

1 Introduction

The paging algorithm employed in a system can have a huge impact on the performance of the system. In this laboratory assignment your task is to examine some of the simplest page replacement algorithms using a simulator.

2 Aim

The aim of this laboratory assignment is to further your understanding of the paging algorithms in question and the problems which have to be solved by this kind of algorithm. It will also serve to examine the following:

- That you understand the different aspects of problems a page-replacement algorithm is designed to solve.
- That you can analyze the algorithms to determine when they perform well and when they perform poorly.

3 Reading Instructions

Before attempting this assignment you should have read Chapter 9 "Virtual-Memory Management" in [1, 2].

The following tools from the UNIX-like command-line can be of use:

- cat(1),
- wc(1), specifically options *-w* and *-l*; and
- grep(1).

Read the manual pages to get an overview of what these commands do.

4 Theory

The source code for the simulator is included with this assignment. The simulator program-file is called `pager`, for the C version located in `cpager/`, or `pypager` for the Python version located in `pypager/`. Both simulators implement the same algorithms and uses pure demand paging.

4.1 The Simulator Written in C

To use the C version you will have to compile it. A `Makefile` is provided to make the process of compilation easier. To compile the program, simply execute the command-line `make pager` in your terminal when your current working directory is `cpager/`. When run, the simulator program reads a page reference string from standard input (stdin) and outputs its actions to standard output (stdout).

To switch between algorithms you should first make changes in one of the source files and then recompile the program. To switch to a different algorithm, edit the file `pager.c` and change the argument to the function `mem_setalg`. The algorithms that are supported by the simulator program can be found in the file `memalg.h`. When done, recompile the program by running `make pager` in the terminal again.

You can also change the number of available pages and frames. This is accomplished by changing the variables `npages` and `nframes` also in `pager.c`.

Running the paging simulator is rather straight-forward. Simply execute it without arguments and it will wait for the page reference string on stdin. The page reference string can be space or new-line separated (or both) and the output will be on the form of one action per line. The page references can optionally be prefixed with either an *r* or a *w*; indicating that the frame should be read from or written to, respectively. If no prefix is supplied, a read

```
$ make pager
[lots of output]
$ cat ../fifo-refstr.txt
r0 w3 r2 r4 w4 w1 r3 r5
$ cat ../fifo-refstr.txt | wc -w
8
$
```

Listing 1: An example from the UNIX-like terminal of how to compute the length of the reference string.

operation is assumed. This means that you could use the reference strings from [1–3] directly, and these would be interpreted as reading operations.

To illustrate all this, the example in Listing 1 illustrates how to easily count the number of page references in our reference string. We see that we have 8 page references in our reference string.

4.2 The Simulator Written in Python

The simulator written in Python 3, named `pypager`, works in exactly the same manner as the one in C, the only difference is the programming language. Since Python code is interpreted no compilation is necessary. Note that the simulator is not backward compatible with Python 2. To run the Python version go to `pypager/` amd run `pager.py`.

To change the algorithm used and number of frames or pages, edit the file `pager.py`. Changing the algorithm used is done by changing the value of the variable `alg`. There are lines prepared for this, just switch to any of the other commented out lines in the code. To change the number of pages or frames, modify the values of the variables `npages` or `nframes`, respectively. Keep in mind that the first page and frame always have the index zero. With three pages in total, you will have page 0, 1, and 2.

Listing 2 illustrates the output of the simulator and how to easily count the number of page faults. Of the 8 page references a total of 6 generated page faults. You may pipe the input into the simulator as in the given example, use redirects, or manually input the data.

The referenced pages in the reference string will generally be prefixed with r for read or w for write, e.g. `r0 w1`. If the prefix is excluded the simulator will assume a read operation, e.g. `0 1` will be interpreted as `r0 r1`.

```
$ cat fifo-refstr.txt | ./pypager/pager.py
page 0 generated page fault
page 0 allocated to free frame 0
page 0 paged in
page 0 mapped to frame 0
page 3 generated page fault
page 3 allocated to free frame 1
page 3 paged in
page 3 mapped to frame 1
page 2 generated page fault
page 2 allocated to free frame 2
page 2 paged in
page 2 mapped to frame 2
page 4 generated page fault
page 4 allocated to free frame 3
page 4 paged in
page 4 mapped to frame 3
page 4 mapped to frame 3
page 1 generated page fault
page 1 allocated to free frame 4
page 1 paged in
page 1 mapped to frame 4
page 3 mapped to frame 1
page 5 generated page fault
page 0 paged out
page 5 paged in
page 5 mapped to frame 0
$ cat fifo-refstr.txt | ./pypager/pager.py | grep "page
    fault" | wc -l
6
```

Listing 2: An example showing a run of the Python simulator.

5 Work

You should start by convincing yourself whether the provided replacement algorithms are correctly implemented. Do this by testing different page reference strings and do the corresponding computations manually using the descriptions of the algorithms given by Silberschatz, Galvin, and Gagne [1–3, ch. 9.4].

Next, you should find one page reference string which adequately shows that the algorithms perform differently in regards to the number of page faults generated for that very string. It is important that you analyze these results in your report. If you can combine this task with the previous, all the better.

Next, you should find one page reference string for each algorithm which yields a minimum page-fault rate for that particular algorithm. The trivial page reference string containing only one repeated page number is not allowed, e.g. 0 0 0 ..., all available pages must be referenced at least once. Naturally you must always include more pages than frames, otherwise the algorithms will never run.

Similarily you should find a page reference string which yields a maximum page-fault rate.

Finally, find proof of Belady's anomaly for the algorithms suffering from it.

The algorithms available are: First-in, first-out (FIFO), Second-chance (SC), and Enhanced second-chance (ESC).

6 Examination

Hand in a written report (in PDF format) describing your results, the conclusions you have drawn from these as well as why you drew these conclusions. In essence, you should do the following:

1. Answer whether the algorithms are correctly implemented or not.
2. Show that the algorithms perform differently for a given reference string. Analyze the results.
3. Provide a page reference string which generates a minimum page-fault rate for each algorithm.
4. Provide a page reference string which generates a maximum page-fault rate for each algorithm.
5. Provide evidence of Belady's anomaly for at least one of the algorithms suffering from it.

You should, as always, provide convincing arguments for your answers and use correct references where applicable. Any references made must include page or section numbers and use the IEEE format.

To make life a bit easier for you a report template is attached, see *report.tex*. (This file is also included in Appendix A.) You may produce your report using any tool, but it should look the same as the one generated by the template. If you use it you can simply execute `make report.pdf` in the terminal to compile the PDF-file.

References

[1] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. 9th ed. International Student Version. Hoboken, N.J.: John Wiley & Sons Inc, 2013.

[2] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. 9th ed. Hoboken, N.J.: John Wiley & Sons Inc, 2013.

[3] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. 8th ed. International Student Version. Hoboken, N.J.: John Wiley & Sons Inc, 2009.

A A Report Template

Here follows the inclusion of a LaTeX report template. This file is also included in the sources as `report.tex`.

```
% $Id: report.tex 1439 2013-11-12 14:29:33Z danbos $
% Author: Daniel Bosk <daniel.bosk@miun.se>
% Date:   17 Dec 2012
\documentclass[a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage[swedish,english]{babel} % english default
    language
\usepackage[hyphens]{url}
\usepackage{hyperref}
\usepackage{prettyref,varioref}
\usepackage[numbers,square]{natbib}
\usepackage{color}
\usepackage{listings}
\bibliographystyle{plainnat}

\title{Laboratory Report on Paging Algorithms}
\author{Your Name}
\date{\today}

%\lstset{style=term}

\begin{document}
\maketitle
\begin{abstract}
  An abstract summarizing the report in no more than 200
      words.
\end{abstract}


\section{Introduction}
\label{sec:Intro}
A gentle introduction to the subject for the reader \dots

Note that this is only a template. Keep the section headers
    and replace all
```

```
33 the content with your own.
34
35
36 \section{Theory}
37 \label{sec:Theory}
38 A short descriptions of the algorithms in question \dots
       with references to
39 \citet{Silberschatz2009osc,Silberschatz2013intl,Silberschatz2013osc},
       and
40 possibly other, reliable sources.
41
42 \section{Method}
43 \label{sec:Method}
44 What simulator and operating system was used. What type of
       sources has been
45 included. In essence, anything related to how the lab was
       performed by you.
46
47 \section{Results}
48 \label{sec:Results}
49 The hard-earned results \dots
50
51 E.g. Reference string X generated Y page-faults of Z
       possible, see the data in
52 \prettyref{app:Data}.
53
54 Genereally speaking the result chapter should be short,
       on-point, and only
55 include objective data. What does this mean? It means only
       your own
56 calculations, and short-hand results from the simulator
       should be presented.
57 The full input and output from the simulator should be put
       inside the appendix
58 at the end of the report, and simply referenced from this
       section. It is also
59 important that subjective statements or analysis takes
       place in this section.
60
61
62 \section{Analysis}
63 \label{sec:Analysis}
64 Analyze and try to explain the results. Compare with your
       own calculations.
65 Motivate your results based on theory and assumptions.
66 Draw conclusions which are then summarized in
       \prettyref{sec:Conclusion}.
67 In this section subjective statements are allowed, but are
       best saved for
68 the conclusions.
69
70
```

```
71
72 \section{Conclusion}
73 \label{sec:Conclusion}
74 Here you summarize the following:
75 \begin{enumerate}
76   \item Answer whether the algorithms are correctly
         implemented or not?
77   \item Show that the algorithms perform differently for a
         given
78   reference string. Analyze the results.
79   \item Provide a reference string which generates a
         minimum page-fault rate
80     for each algorithm.
81   \item Provide a reference string which generates a
         maximum page-fault rate
82     for each algorithm.
83   \item Provide evidence of Belady's anomaly for at least
         one of the
84   algorithms suffering from it.
85 \end{enumerate}
86 Remember that you should never present any new data in your
     conclusions.
87 \textbf{In this section you merely draw conclusions based
     on your results and your own analysis from the previous
     sections}.
88
89
90 \bibliography{literature}
91
92 \appendix
93
94 \section{Data}
95 \label{app:Data}
96 Reference strings and corresponding output are given here.
97 The first FIFO-experiment yielded the following data.
98 The reference string:
99 \lstinputlisting{fifo-refstr.txt}
100 Inputted using the command \dots
101
102 \noindent And the resulting output from the program:
103 \lstinputlisting{fifo-result.txt}
104
105 \end{document}
```

Here follows a compiled version of the above code.

Laboratory Report on Paging Algorithms

Your Name

January 4, 2017

Abstract
An abstract summarizing the report in no more than 200 words.

1 Introduction

A gentle introduction to the subject for the reader ...

Note that this is only a template. Keep the section headers and replace all the content with your own.

2 Theory

A short descriptions of the algorithms in question ... with references to Silberschatz et al. [1, 2, 3], and possibly other, reliable sources.

3 Method

What simulator and operating system was used. What type of sources has been included. In essence, anything related to how the lab was performed by you.

4 Results

The hard-earned results ...

E.g. Reference string X generated Y page-faults of Z possible, see the data in A.

Genereally speaking the result chapter should be short, on-point, and only include objective data. What does this mean? It means only your own calculations, and short-hand results from the simulator should be presented. The full input and output from the simulator should be put inside the appendix at the end of the report, and simply referenced from this section. It is also important that subjective statements or analysis takes place in this section.

5 Analysis

Analyze and try to explain the results. Compare with your own calculations. Motivate your results based on theory and assumptions. Draw conclusions which are then summarized in Section 6. In this section subjective statements are allowed, but are best saved for the conclusions.

6 Conclusion

Here you summarize the following:

1. Answer whether the algorithms are correctly implemented or not?
2. Show that the algorithms perform differently for a given reference string. Analyze the results.
3. Provide a reference string which generates a minimum page-fault rate for each algorithm.
4. Provide a reference string which generates a maximum page-fault rate for each algorithm.
5. Provide evidence of Belady's anomaly for at least one of the algorithms suffering from it.

Remember that you should never present any new data in your conclusions. **In this section you merely draw conclusions based on your results and your own analysis from the previous sections**.

References

[1] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. John Wiley & Sons Inc, Hoboken, N.J., 8 edition, 2009. International Student Version.

[2] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. John Wiley & Sons Inc, Hoboken, N.J., 9 edition, 2013. International Student Version.

[3] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. John Wiley & Sons Inc, Hoboken, N.J., 9 edition, 2013.

A Data

Reference strings and corresponding output are given here. The first FIFO-experiment yielded the following data. The reference string:

```
r0 w3 r2 r4 w4 w1 r3 r5
```

Inputted using the command ...
And the resulting output from the program:

```
page 0 generated page fault
page 0 allocated to free frame 0
page 0 paged in
page 0 mapped to frame 0
page 3 generated page fault
page 3 allocated to free frame 1
page 3 paged in
```

```
page 3 mapped to frame 1
page 2 generated page fault
page 2 allocated to free frame 2
page 2 paged in
page 2 mapped to frame 2
page 4 generated page fault
page 4 allocated to free frame 3
page 4 paged in
page 4 mapped to frame 3
page 4 mapped to frame 3
page 1 generated page fault
page 1 allocated to free frame 4
page 1 paged in
page 1 mapped to frame 4
page 3 mapped to frame 1
page 5 generated page fault
page 0 paged out
page 5 paged in
page 5 mapped to frame 0
```