CS 3013 Operating Systems

WPI, C Term 2007

Project 5 – Memory Management and Performance

This project is to be done by each individual student. It is NOT a group project

Project Description

The primary purpose of this project is to compare the performance of standard file I/O using the read() system call for input with memory-mapped I/O where the mmap() system call allows the contents of a file to be mapped to memory. Access to the file is then controlled by the virtual memory manager of the operating system. In both cases, you will need to use the open() and close() system calls for opening and closing the file for I/O.

For the project, you should write a program proj5 that takes a file name as command-line argument and computes the percentage of printable characters in the file. To do so you should use two routines: isprint(), which determines if a byte value is a printable character; and isspace(), which determines if a byte is a space, newline, tab, etc. Check the man pages of these routines for details and the needed include file.

The only output from the program should be two lines with the number of printable characters in the file, the number of whitespace characters in the file, the total number of bytes in the file and a percentage printed as an integer (rounded down to the next lowest percentage point) between 0 and 100 such as:

```
% proj5 proj5.c
5730 printable  characters out of 5874 bytes,  97%
1546 whitespace characters out of 5874 bytes,  26%
```

The default behavior of the program should be to read bytes from the file in chunks of 1024 bytes using the *read()* system call. However your program should have an optional second argument that controls the chunk size for reading or to tell the program to use memory-mapped file I/O. In the latter case your program should map the entire contents of the file to memory. The syntax of your program:

```
proj5 srcfile [size|mmap]
```

where `srcfile` is the file on which to determine the percentage of printable and whitespace characters. If the optional second argument is an integer then it is the `size` of bytes to use on each loop when reading the file using the *read()* system call. Your program should enforce a chunk size limit of no more than 8192 (8K) bytes. Your program should traverse the buffer of bytes read on each iteration and keep track of the number of bytes and printable characters.

If the optional second argument is the literal string "mmap" then your program should not use the read() system call, but rather use the mmap() system call to map the contents of srcfile to memory. You should look at the man pages for mmap() and munmap() as well as the sample program mmapexample.C for help in using these system calls. Once your program has mapped the file to memory then it should iterate through all bytes in memory to count printable characters. You should verify that the file I/O and memory mapped options of your program show the same output for the same file as a minimal test of correctness.

Performance Analysis

Once you have your program functionally working for both types of I/O then you need to perform an analysis to see which type of I/O works better for different size files. For this portion of the project, you should reuse the first part of Project 1, which allows you to collect system usage statistics. The two usage statistics of interest for this project are (hard and soft) page faults and the total response (wall-clock) time. A sample invocation of your proj5 program on itself using doit with the largest read size would be the following where proj5 prints its output and then doit prints the resource usage statistics for the program.

```
% proj5 proj5.c 8192
5730 printable  characters out of 5874 bytes,  97%
1546 whitespace characters out of 5874 bytes,  26%
< resource usage statistics for proj5 process >
```

At the minimum, you must test your program running under five configurations for input files of different sizes. The five configurations are standard file I/O with read sizes of 1, 1K, 4K, and 8K bytes as well as with memory mapped I/O. You should determine performance statistics for each of these configurations on a variety of file sizes. As an aid to finding a range of file sizes, the directory `/var/lib/rpm` on the CCC machines has some large files such as `Filemd5s` or `Packages`. You should look for other files with a range of sizes.

Once you have executed your program with different configurations on a range of files, you should plot your results on two graphs where the file size is on the x-axis and the system statistic of interest (major page faults or wall-clock time) is on the y-axis. Each graph should have one line for the results of each configurations.

You should include these graphs as well as a writeup on their significance in a short (1-2 pages of text) report to be submitted along with your source code. You should indicate which configurations clearly perform better or worse than others on a given performance metric and whether there is clearly a "best practice" technique to use.

Submission of Project

Use turnin to turn in your project using the assignment name "proj5". You should submit the source code for your program and a soft copy of your report.

CS3013 – Operating Systems
Project 5 Evaluation Sheet

Name:

Submission Mechanics	**15 total**
You used turnin and followed the rules given above, thus saving the TA's considerable time and effort.	5 points
When we cd into proj5, we can type "make" and the executable is produced from the source(s).	5 points
The code has comments and is structured – it's not spaghetti code.	5 points
Basic Commands	**45 total**
Output format is as specified in the write-up.	15 points
We have a number of scripts we will run. These scripts will examine that your code is behaving the way it should. These scripts will try files of various sizes as well as the configurations discussed in the writeup.	
We are unable to break your code using various input parameters.	10 points
Student program gives the correct values for printable and whitespace characters.	20 points
Performance Analysis	**40 total**
Various configurations and file sizes are analyzed.	15 points
Graphs that make sense are submitted.	15 points
The performance results are stated in a clear written fashion	10 points