CS 446 Homework 2

**Chapter 3**

1. [Linux Operating System] ***free*** is a command that displays used and available memory in your system. Read man page of ***free*** command. Run the command ***free –o*** several times, running other programs in between, and store the results in a file. Draw a graph as follows: X-axis: MB-used; for the Y-axis, use (i) Memory Used per unit time; (ii) (Memory Used – Memory Buffered – Memory Cached) per unit time; and (3) Swap Used per unit time. Explain the behavior of this graph with respect to memory utilization in the presence of running various applications.
2. Memory Used per unit time
   1. *Data points taken every 15 seconds*
3. (Memory Used – Memory Buffered – Memory Cached) per unit time
   1. *Data points taken every 15 seconds*
4. Swap Used per unit time
   1. *Data points taken every 15 seconds*

**As the presence of more application appears (three new firefox instances every 15 seconds), memory usage increases in comparison to the available free memory left on the system . In addition, both the buffer and cache slowly increase in conjunction with pages filling up the buffer cache. On the other hand, swap remains at zero as the RAM never reaches a state where it is full and must dedicate a partition on the HDD/SDD to use as extra memory space.**

1. [Any System] Plot a histogram and calculate the mean and median of the sizes of the executable binary files on a computer to which you have access. On a Windows system, look at all .exe and .dll files; on a UNIX system look at all executable files in */bin*, */usr/bin*, and */local/bin* that are not scripts (or use the file utility to find all executables). Determine the optimal page size for this computer just considering the code (not data). Consider internal fragmentation and page table size, making some reasonable assumption about the size of a page table entry. Assume that all programs are equally likely to be run and thus should be weighted equally.

**Windows 🡪 \*.exe files**

A screenshot of a computer

Description automatically generated

**Windows 🡪 \*.dll files**

A screenshot of a computer

Description automatically generated

**Results - Windows Compiled Files Data**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **# Of Files** | **File Size (GB)** | **File Size (MB)** |
| **\*.exe** | 6959 | 1.26 | 1260 |
| **\*.dll** | 47683 | 12.3 | 12300 |

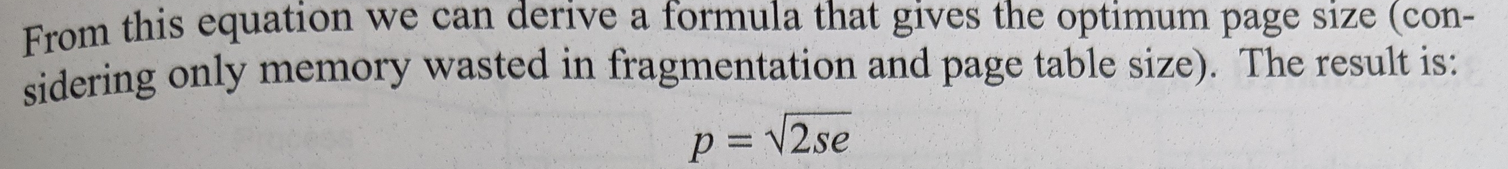
Table

Description automatically generated

**Windows DLL/Exe Files 🡪 Mean and Median**

**Optimum Page Size**

Graphical user interface, text

Description automatically generated

***s = Average Process Size (bytes)***

***e = Page Table Size (MB)***

**s (exe files) = .077 MB and e = 6144 MB**

**p = = 30.76 MB**

**Chapter 2**

1. [Linux Operating System] ***df*** is a command that displays the amount of disk space available on the file system containing each file name argument. Read man page of ***df*** command. Run the command ***df*** to find out how many disk blocks are available and how many are in use. Does the sum of these equals the total number of disk blocks on the disk? If not, explain why there is a difference. Next run the command ***df –i*** to find out how many i-nodes are available and in use. Now create a new file with just a few characters in it, and again run ***df*** and ***df –i*** commands. Explain the effect of creating this new file. Now increase the size of this new file by entering a large number (> 5000) of characters, and again run ***df*** and ***df –i*** commands. Explain the effect of increasing the size of the new file.

**Df Command**

A black screen with white text

Description automatically generated with low confidence

Root partition

* ***Available: 5321620; Used: 2433012***
* ***Sum (available + used) = 7,754,632***
* ***Sum of available and used disk blocks does not equal the total number of disk blocks because the difference in space is reserved for critical processes in which if the system is 100% full, there will always be space left for these critical processes to keep the system running (essentially a built-in safety measure).***

**Df -i Command**

Text

Description automatically generated with medium confidence

Root partition

* ***Total Inodes: 524288 🡪 IUsed: 64626 | IFree: 459662***

**Df Command – New File with few characters**

Text

Description automatically generated with medium confidence

* ***Available: 5321612 | Used: 2433020***
* ***Total Inodes: 524288 🡪 IUsed: 64628 | IFree: 459660***
* ***By creating the new file with a few characters in it, available/free space decreased with its respective used space increasing by the same value (essentially disk space was used up by creating the file). In addition, since a new file was created, an inode was created to store the metadata of the file, hence the increase in “IUsed” and decrease in “IFree”. Every file has one inode, and if you run out of inodes, you will not be able to create a new file, even though there is still space left on your system.***

**Df -i Command – New File with large number of characters**

Text

Description automatically generated

Random Character Generator (10000 characters)

Character Check (10000 characters)

* ***Available: 5321600 | Used: 2433032***
* ***Total Inodes: 524288 🡪 IUsed: 64629 | IFree: 459659***
* ***By creating the new file with a few characters in it, available disk space decreased with its respective used space increasing by the same value (essentially disk space was used up by creating the file, however this time, more disk space was used as the file is bigger). In addition, since a new file was created, an inode was created to store the metadata of the file, hence the increase in “IUsed” and decrease in “IFree”. Every file has one inode, and if you run out of inodes, you will not be able to create a new file, even though there is still space left on your system.***

1. [Programming Problem] Write a program that starts at a given directory and descends the file tree from that point, recording the sizes of all the files it finds. When the traversal is complete, the program should print a histogram of the file sizes using a bin width specified as a parameter into the program (e.g., with 1024, file sizes of 0 to 1023 to in one bin, 1024 to 2047 go in the next, etc.

* **Utilized Freedman-Diaconis rule in determining the perfect width to use for the histogram.**
  + **Benefits: Faster plotting and better plots (less cluttered)**

**Histogram Plot (Freedman-Diaconis Rule) 🡪 Bin Width: 385779 | Bin Count: 150**

Chart, histogram

Description automatically generated

**Histogram Plot (Static Bin Width) 🡪 Bin Width: 1024 | Bin Count: 56281**

Chart, histogram

Description automatically generated