Modern Operating Systems | (4th Edition)

See this solution in the app

Chapter 3, Problem 50P | Bookmark | Show all steps: ON

Problem

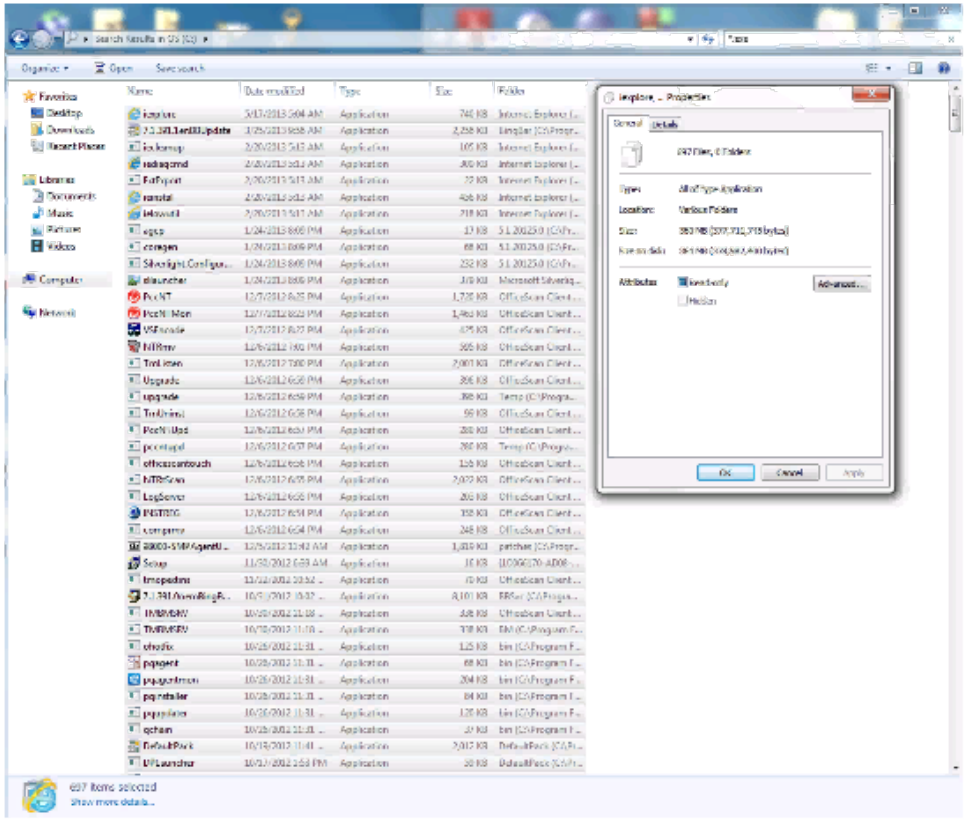
Plot a histogram and calculate the mean and median of the sizes of executable binary files on a computer to which you have access. On a Windows system, look at all .exe and .dll files; on a UNIX system look at all executable files in */bin*, */usr/bin*, and */local/bin* that are not scripts (or use the *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.

Step-by-step solution

Step 1 of 7

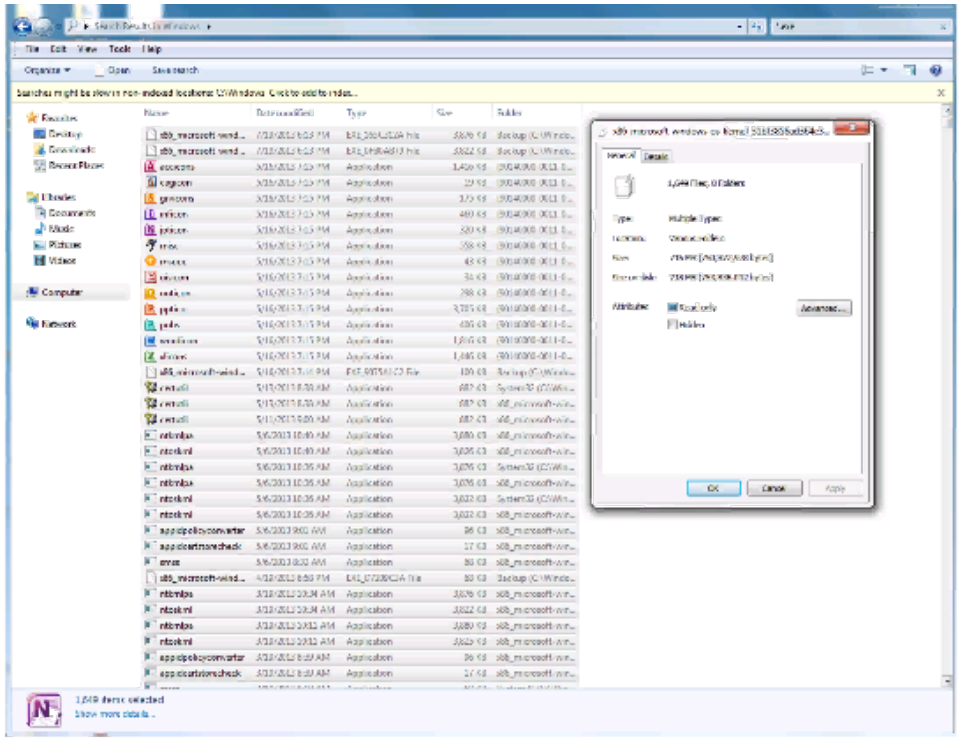
Windows Operating System allows the executable files (.exe) and the direct linked library (.dll) files for execution of the applications and system internal processes. The count and total size of the (.exe) files in the windows folders (C:\ and C:\windows) are shown in the below snapshot:

Windows folder (C:\)



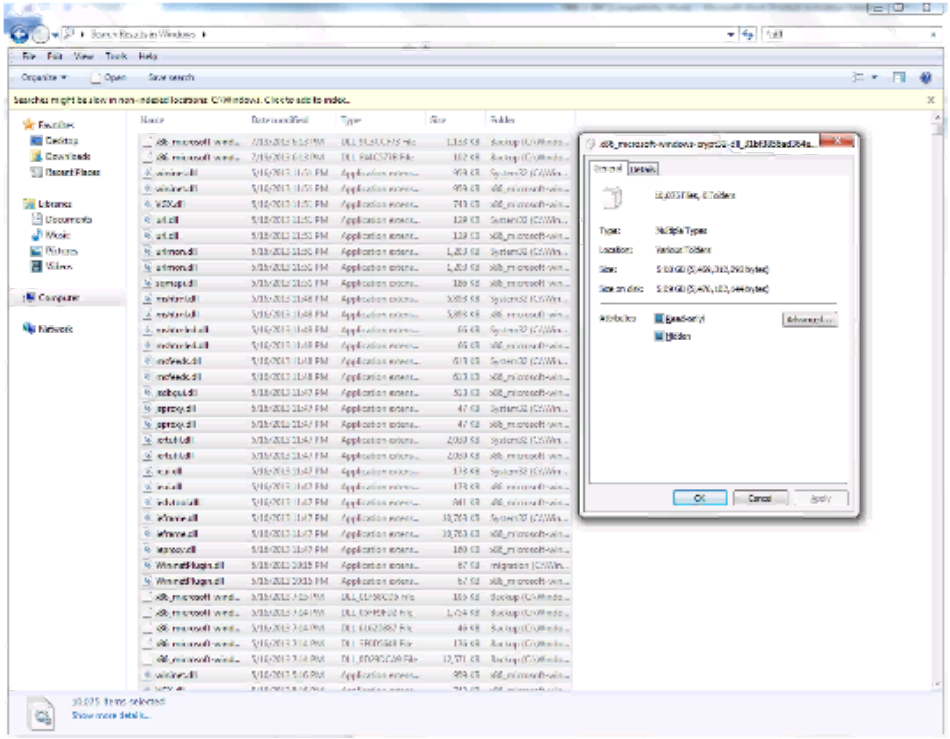
Comment

Step 2 of 7



Comment

Step 3 of 7

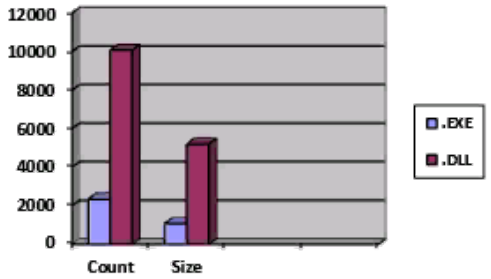


Comment

Step 4 of 7

File Type	File Count	File Size (MB)
.exe	2,346	1076
.dll	10,075	5202

The graphical representation for the above data is as follows:



Comment

Step 5 of 7

When an application is executed, then a corresponding process is started. This process in order to save on the usage of RAM adapts the paging. In the paging technique, memory is not allocated to the complete application, but part of the application is given the RAM, rest of the application is executed by constantly getting the needed part of application into the RAM. This concept is also known as the virtual memory.

Page is fixed unit size into which the virtual memory for implementation of paging is divided. The page is stored in the RAM whereas is corresponding detailed memory block; page frame; is stored in the physical memory. The details of the page frame are stored in the page table, which contains the memory segment used, the offset values and other details for the same. In a 16-bit virtual memory address, first 4 bits are used to store the page number and rest 12 bits are used to maintain the offset for the file. This allows loading 16 pages each having 4 MB data to be loaded in the virtual memory.

Comment

It has been noticed that almost 50% times the pages loaded in the virtual memory do not contain any text or data, but are blank; creating internal fragments. If n segments are stored in the memory, each having a fixed page size of p bytes, then a total of $(n \times p)/2$ bytes is held in the internal fragments created.

The average size of each process is s bytes, stored with fixed page size of p bytes. The page entry to be stored in the page table uses e bytes each. Hence the count of pages created per process is s/p and the size of page table is $(s \times e)/p$ bytes. The internal fragmentation created is of size $p/2$ (assuming 50%).

Hence, the total overhead created is: $overhead = (s \times e)/p + p/2$

Comment

Step 7 of 7

If the page size is small, then it increases the size of the page table due to count of entries and if the page size is large then it affects the internal fragmentation size. Hence the optimal page size is: $p = \sqrt{2se}$

The average size of each process is:

$$(total\ memory\ size)/(process\ count) = (1076 \times 1024)/2346 = 469.6$$

The page table for the machine is 512 KB.

Hence, the optimal page size for the machine is:$= \sqrt{2 \times 469.6 \times 512} = 693.5 \approx 694$ KB.

Comment

Was this solution helpful? 1 0

Chapter 3, Problem 24P

A machine has 48-bit virtual addresses and 32-bit physical addresses. Pages are 8 KB. How many entries are needed for a single-level linear page table?

See solution

Chapter 3, Problem 27P

Suppose that the virtual page reference stream contains repetitions of long sequences of page references followed occasionally by a random page reference. For example, the sequence: 0, 1, … , 511, 431, 0, 1, … , 511, 332, 0, 1, … consists of...

See solution