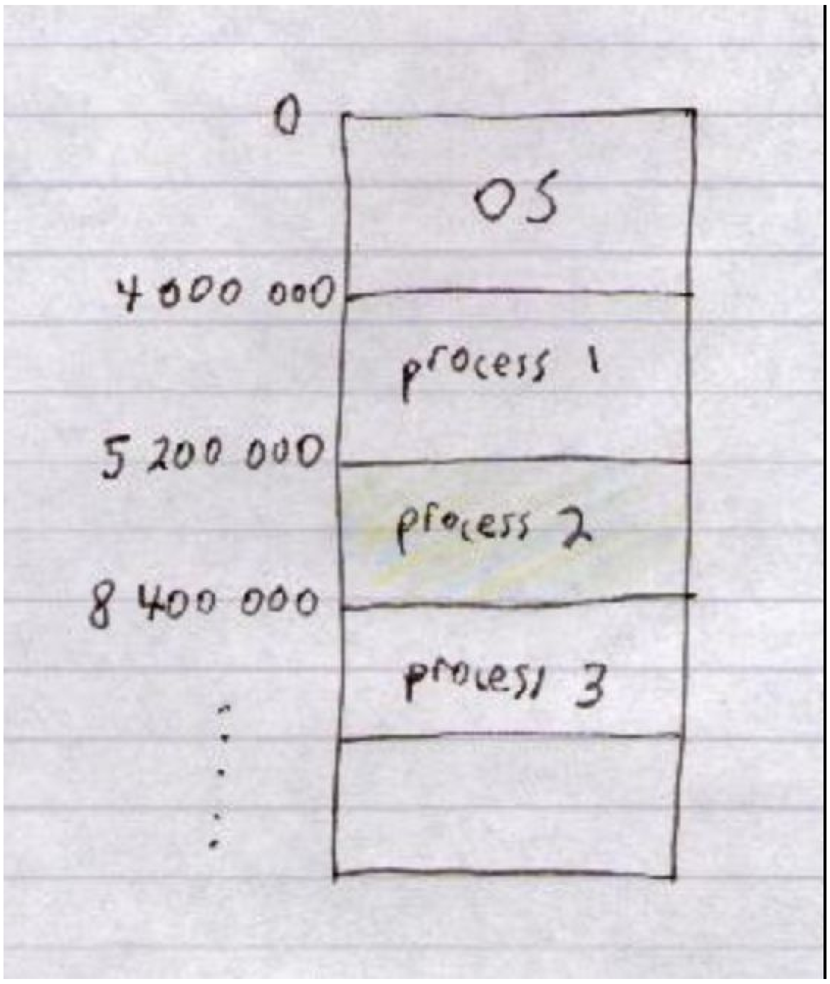
# CA170: Week 11

## Memory

### Memory Protection

- Basic idea is that there is such a thing as a "bad" memory access.
- User process runs in a different memory space to OS process (and other processes).
- On multi-process system (whether single-user or multi-user) each process has its own memory space in which (read-only) code and (read-write) data live.
    - i.e. Each process has defined limits to its memory space:

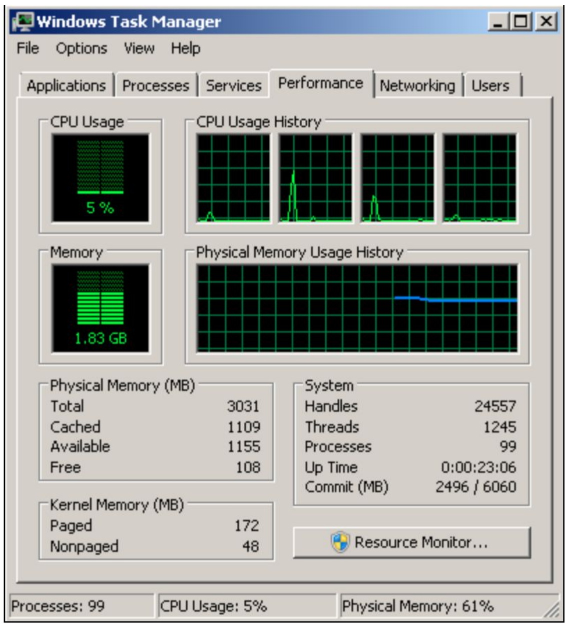


- Q. Even if there are no malicious people, process memory areas need to be protected from each other - Why?
- Above, each process has two registers - a base register and a limit register.
    - e.g. The base register for process 2 has contents 5200000. Its limit register has contents 3200000.
    - And then every memory reference is checked against these limits.
    - A simple model would look like this:

```
When reference is made to memory location x:

if x resolves to between base and (base+limit)
  return pointer to memory location in RAM
else
  OS error interrupt
```

- Q. Load values into base or limit registers are privileged instructions. Why?
    - As we shall see, in fact memory protection has evolved far beyond this simple model.

## Map of memory (Task Manager)



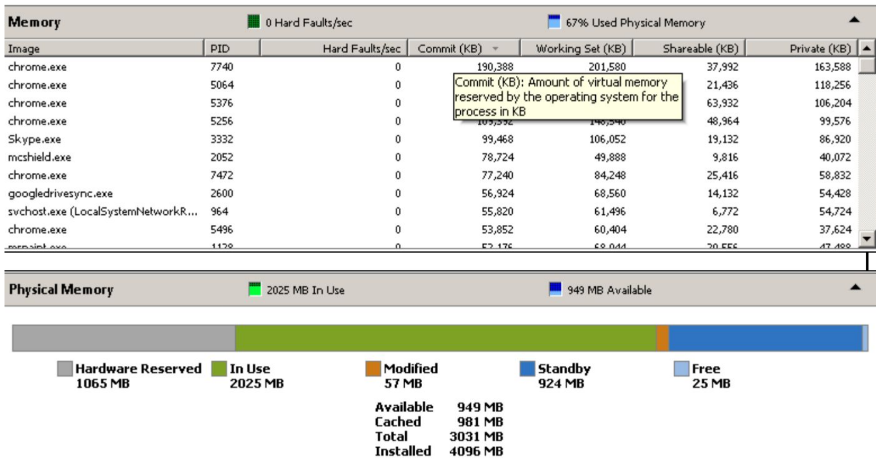
| Physical Memory (MB) | |
|---|---|
| Total | 3031 |
| Cached | 1109 |
| Available | 1155 |
| Free | 108 |

| Kernel Memory (MB) | |
|---|---|
| Paged | 172 |
| Nonpaged | 48 |

| System | |
|---|---|
| Handles | 24557 |
| Threads | 1245 |
| Processes | 99 |
| Up Time | 0:00:23:06 |
| Commit (MB) | 2496 / 6060 |

Processes: 99 CPU Usage: 5% Physical Memory: 61%

- Looking at memory use on Windows in Task Manager.



**Memory** 0 Hard Faults/sec 67% Used Physical Memory

| Image | PID | Hard Faults/sec | Commit (KB) | Working Set (KB) | Shareable (KB) | Private (KB) |
|---|---|---|---|---|---|---|
| chrome.exe | 7740 | 0 | 190,388 | 201,580 | 37,992 | 163,588 |
| chrome.exe | 5064 | 0 | [illegible] | [illegible] | 21,436 | 118,256 |
| chrome.exe | 5376 | 0 | [illegible] | [illegible] | 63,932 | 106,204 |
| chrome.exe | 5256 | 0 | [illegible] | [illegible] | 48,964 | 99,576 |
| Skype.exe | 3332 | 0 | 99,468 | 106,052 | 19,132 | 86,920 |
| mcshield.exe | 2052 | 0 | 78,724 | 49,888 | 9,816 | 40,072 |
| chrome.exe | 7472 | 0 | 77,240 | 84,248 | 25,416 | 58,832 |
| googledrivesync.exe | 2600 | 0 | 56,924 | 68,560 | 14,132 | 54,428 |
| svchost.exe (LocalSystemNetworkR... | 964 | 0 | 55,820 | 61,496 | 6,772 | 54,724 |
| chrome.exe | 5496 | 0 | 53,852 | 60,404 | 22,780 | 37,624 |

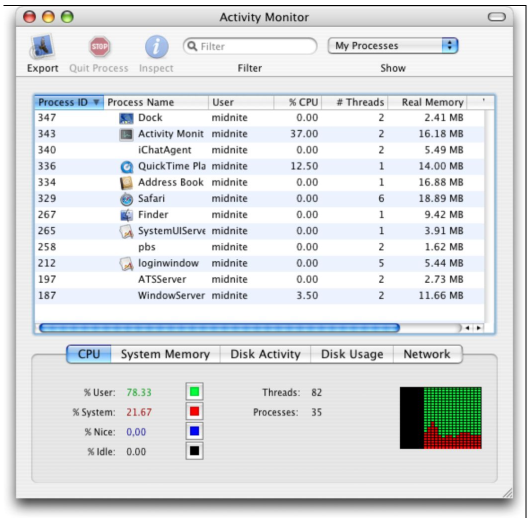
Commit (KB): Amount of virtual memory reserved by the operating system for the process in KB

**Physical Memory** 2025 MB In Use 949 MB Available

| Hardware Reserved | In Use | Modified | Standby | Free |
|---|---|---|---|---|
| 1065 MB | 2025 MB | 57 MB | 924 MB | 25 MB |

| | |
|---|---|
| Available | 949 MB |
| Cached | 981 MB |
| Total | 3031 MB |
| Installed | 4096 MB |

- Click "Resource Monitor" in the above and you get further breakdowns. (See "Memory" tab)
- Hover over items to see definitions.
- "Standby" here (in Windows 7) is really free RAM. See explanation later.
- See Processes (ps) in Linux/Unix



| Process ID | Process Name | User | % CPU | # Threads | Real Memory |
|---|---|---|---|---|---|
| 347 | Dock | midnite | 0.00 | 2 | 2.41 MB |
| 343 | Activity Monit | midnite | 37.00 | 2 | 16.18 MB |
| 340 | iChatAgent | midnite | 0.00 | 2 | 5.49 MB |
| 336 | QuickTime Pla | midnite | 12.50 | 1 | 14.00 MB |
| 334 | Address Book | midnite | 0.00 | 1 | 16.88 MB |
| 329 | Safari | midnite | 0.00 | 6 | 18.89 MB |
| 267 | Finder | midnite | 0.00 | 1 | 9.42 MB |
| 265 | SystemUIServe | midnite | 0.00 | 1 | 3.91 MB |
| 258 | pbs | midnite | 0.00 | 2 | 1.62 MB |
| 212 | loginwindow | midnite | 0.00 | 5 | 5.44 MB |
| 197 | ATSServer | midnite | 0.00 | 2 | 2.73 MB |
| 187 | WindowServer | midnite | 3.50 | 2 | 11.66 MB |

| | | | |
|---|---|---|---|
| % User: | 78.33 | Threads: | 82 |
| % System: | 21.67 | Processes: | 35 |
| % Nice: | 0.00 | | |
| % Idle: | 0.00 | | |

- 
- Apple OS X has Activity Monitor.
- OS X is Unix family.
- You can also use ps on the Apple Terminal (command-line).
- Image from here. Creative Commons.

## Compile-time, Load-time and Run-time

- The transformations made to the program from the code written by a human to instructions executed by the CPU.
- Compiled program (e.g. typical use of C++, Java):
    - Program typically written "in factory" in HLL, with comments, English-like syntax and variable names, macros and other aids to the programmer.
    - Program is then compiled "in factory" into an efficient, machine-readable version, with all of the above stripped, optimisations made, and everything at "compile-time" translated and resolved as much as possible, so as little as possible needs to be done when it is run.
    - At different times, on different machines, and with different other programs already running, the user will "launch" or "run" a copy of this compiled program. Any further translation that could not be done at compile-time will now be done at this "load-time". Then the algorithm itself actually starts to run.
    - Any change that has to be done while the algorithm has already started to run, is a change at "run-time".
- Interpreted program (e.g. typical use of Javascript, Shell):
    - No compilation step. Source code itself is read at "load-time".

## Memory mapping (or binding)

- Consider what happens with the memory allocation for a program's global variables.
- Programmer defines global variable "x". Refers to "x" throughout his High-Level Language code:

```
do 10 times
    print(x)
    x := x+7
```

- Clearly when the program is running, some memory location will be set aside for x and the actual machine instructions will be:

```
do 10 times
    read contents of memory location 7604 and print them
    read contents of loc 7604, add 7, store result back in loc 7604
```

- How "x" maps to "7604" can happen in many ways:
    1. Source code: The programmer maps x to 7604 in the source code.
    2. Compile-time: If x is mapped to 7604 at this point (or before) then the program can only run in a certain memory location. (e.g. DOS - single user, single process)
    3. Load-time: The compiler might map x to "start of program + 604". Then when the program is launched, the OS examines memory, decides to run the program in a space starting at location 7000, resolves all the addresses to absolute numerical addresses, and starts running.
    4. Run-time: Even after the program has started, we may change the mapping (move the program, or even just bits of it, in memory), so that next time round the loop it is:

       ```
       read contents of memory location 510 and print them
       read contents of loc 510, add 7, store result back in loc
       510
       ```

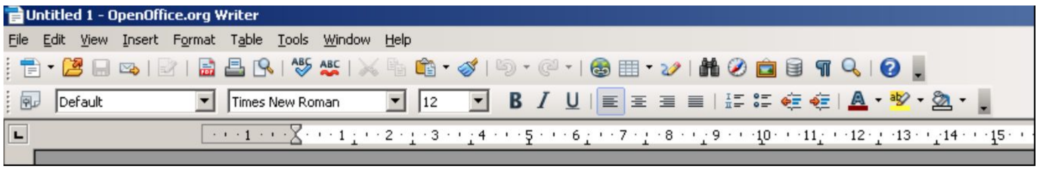
       Obviously, OS has to manage this!
       User can't. Programmer can't. Even Compiler designer can't.

## Questions

- Question - Why is it not practical for the OS to say to a user on a single-user system: "I'm sorry, another program is occupying memory space 7000-8000. Please terminate the other program before running your program"
- Question - Why is it not practical for the OS to say to a user on a multi-user system: "I'm sorry, another program is occupying memory space 7000-8000. Please wait until the other program terminates before running your program"
- Question - Why can't program writers simply agree on how to divide up the space? Some programs will take memory locations 7000-8000. Other programs will take locations 8000-9000, etc.
- Question - The user's program has started. The OS needs to change what memory locations things map to, but binding is done at load-time. Why is it not practical for the OS to say to the user: "Please reload your program so I can re-do the binding"
- Question - Why is it not practical to say to a user: "Please recompile your program"

## Partial load of programs (desired)

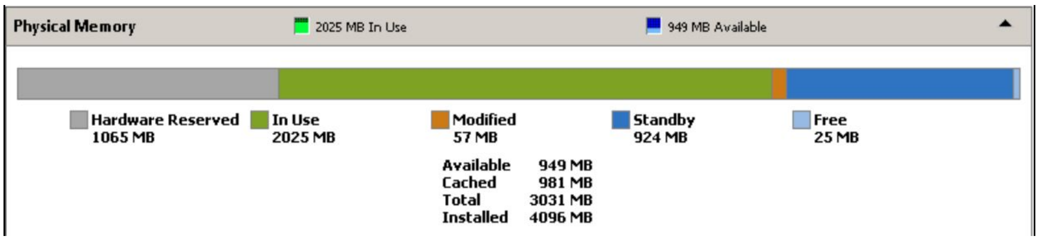
- Consider: Does the whole program have to be loaded into memory when it is run?
- Many applications contain vast amounts of functionality that is rarely or never used.
- e.g. Run web browser for years. Never once click on some menu options.
- But was the code behind those menu options loaded into memory every single time, for years?
- How can we load only part of a program?


Untitled 1 - OpenOffice.org Writer
File Edit View Insert Format Table Tools Window Help
Default
Times New Roman
12

- Many applications contain functionality that is rarely or never used.

## Pre-load programs and keep old ones in memory (desired)

- In apparent contrast to only loading part of a program into RAM before starting execution is the following idea, where we pre-load things (and keep old things) in memory that we may not even need



Physical Memory    2025 MB In Use    949 MB Available

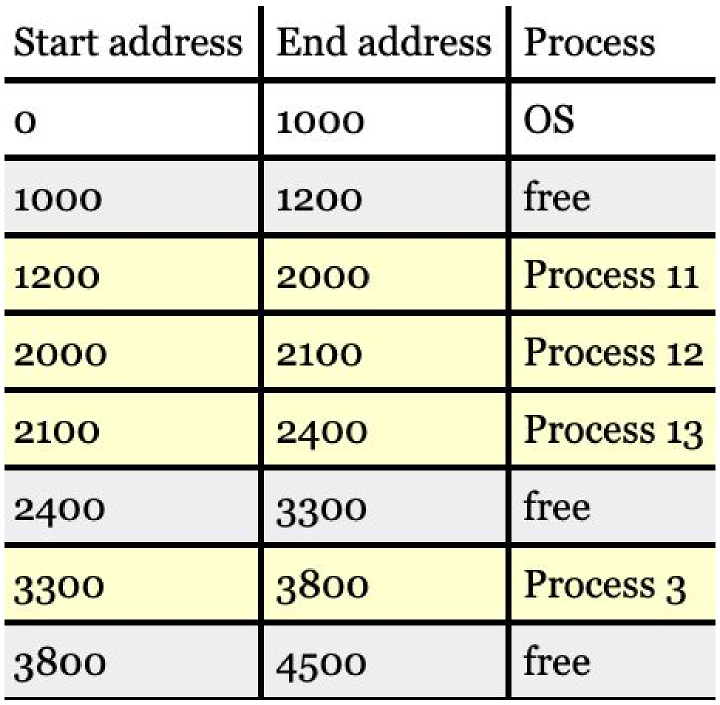
| Hardware Reserved | In Use | Modified | Standby | Free |
|---|---|---|---|---|
| 1065 MB | 2025 MB | 57 MB | 924 MB | 25 MB |

```
Available    949 MB
Cached       981 MB
Total       3031 MB
Installed   4096 MB
```

- Above we saw this typical memory map of a Windows 7 system.
- We saw there is not much free memory, but I said that "Standby" is really free memory.
- I will now explain this.
    - In the "Standby" section, the OS has filled RAM with stuff we may not even need:
    1. It keeps old code and data that may be needed again.
    2. It may pre-load commonly used programs into memory in case they are run.
        - See Windows Prefetch and SuperFetch algorithms.
        - Info on recently-run programs stored in .PF files in something like C:\Windows\Prefetch
- This is a good idea. First, this RAM is actually still free for use if needed. It is not being reserved. When more RAM needed, it can be taken from here.
- Second, pre-loads can be done when the system is not busy and need not detract from the user experience. Keeping old data in RAM of course costs nothing.
- Third, if the old data is needed again, or if those common programs are run, they are already in RAM and will run straight away without reading from disk. If they are not needed, there is no problem.
- Why fill RAM with stuff we may not need? For speed. This is the "unused RAM is wasted RAM" idea.

## Contiguous memory allocation

- Memory management
- Processes are large. Take up a lot of memory.
- Imagine if this has to be contiguous - all in one unbroken chunk from memory location A to location A+n.
- End up with a memory map like this:



| Start address | End address | Process |
|---|---|---|
| 0 | 1000 | OS |
| 1000 | 1200 | free |
| 1200 | 2000 | Process 11 |
| 2000 | 2100 | Process 12 |
| 2100 | 2400 | Process 13 |
| 2400 | 3300 | free |
| 3300 | 3800 | Process 3 |
| 3800 | 4500 | free |

- Problems with contiguous memory spaces:
    1. Programs starting and ending all the time. End up all over memory. Will have gaps everywhere.
    2. When load program, have to find contiguous space big enough to hold it.
    3. "Holes" develop in memory. Lots of memory free but not contiguous so can't be used.
    4. When process asks for more memory at run-time, might not be any free above it. (See Process above.)
    5. Difficult to load only part of the program at startup as opposed to whole thing (must reserve big amount of space above/below it).
    6. Difficult to swap out only part of a process as opposed to whole thing, unless we have some mechanism for identifying parts of a process.

## Paging

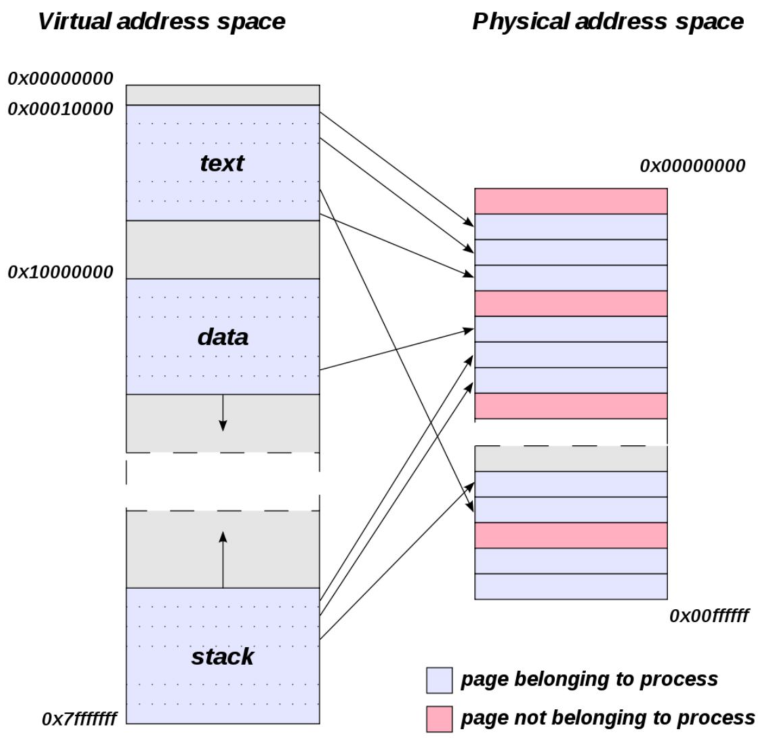
- The modern OS does Paging.
- Non-contiguous memory spaces.
- Program broken (automatically by OS) into lots of small chunks that flow like water into every crack and gap in memory, and out to disk (swapping) and back in.

## Memory mapping with paging

- Program logical (or virtual) memory space is divided into pages.
- Pages are of fixed size, say 4 K.
- On Linux can query page size at command line with getconf:

```
$ getconf PAGESIZE
4096
```

- Some architectures support multiple page sizes (pages, large pages, huge pages).
- A program references a memory location in its virtual address space as an offset within a particular page.
- Physical memory is divided into frames, of same size as pages, so that pages can be loaded into the frames.
- Program loaded into memory. Each page is loaded into a frame. Crucially, the frame can be anywhere in memory. Any free frame will do.
- Page table shows where the program is.



- Paging means the program can be physically located all over memory.
- From here.
- Example
- e.g. Program has logical memory space: page 0, .., page 3. Physical memory frame 0, .., frame 7. Page table:

```
page -> frame
0 -> 1
1 -> 4
2 -> 3
3 -> 7
```

- Don't have to be contiguous, don't even have to be in order.
  Page 0, offset 2,   maps to:   Frame 1, offset 2,
  Page 3, offset n,   maps to:   Frame 7, offset n,
  etc.

## Paging advantages

- Holes and fragmented memory are no problem.
- When load program, do not need to search for contiguous space, only enough memory in different locations.
- If a bit of memory is available anywhere, program can use it.
- Can easily ask for more memory, in any location.
- Can easily load only parts of program.
- Can easily swap only parts of process to disk.
- Makes maximal use of memory. Indeed, we would find modern machines almost unusable without paging.
- Separation of process memory from other processes. Process cannot address memory it doesn't own. Has no language to even describe such memory.

- Paging disadvantages:
    - Overhead - have to resolve addresses at run-time.

## Paging and Swapping (Page faults)

- Swapping pages out to disk:
- Program addresses a logical space. This logical space maps to pages scattered all over memory and on disk. Much of program can be left on disk if not needed. OS hides this from user and programmer.
- Only need: Active processes in memory.
- In fact, only need: Active bits of active processes in memory.
- OS decides what bits of programs stay in memory at any given time.
- If page not in memory it is not an error, OS just fetches it. This is called a Page fault.
- Not really: Disk as extra memory.
    - Disk I/O is slow. So using disk just as "extra memory" is not really the idea.
    - Yes it is extra memory, but program runs much slower if too much disk I/O.
    - Recall Memory hierarchy.
- More like: Disk as overflow of memory.
    - What we really use disk for is as an overflow/buffer/cache of memory. Use it so we don't have to worry about the exact size of memory - worry that launching the next program will cause OS to crash. Instead the system slows down gracefully as we run more progs. (More page faults, more disk I/O, system slows, user closes some programs.)
    - Without using disk as overflow, when we run out of memory there will be sudden and catastrophic crash of the system

## Major and minor page faults

- Minor (soft) page fault - Page is actually in RAM though was not marked as part of the "working set" for this process.
- e.g. Page was pre-fetched in case needed.
- e.g. Page was brought into RAM for another process.
- e.g. Old page marked free but not overwritten.
- Can resolve minor page fault without disk read.

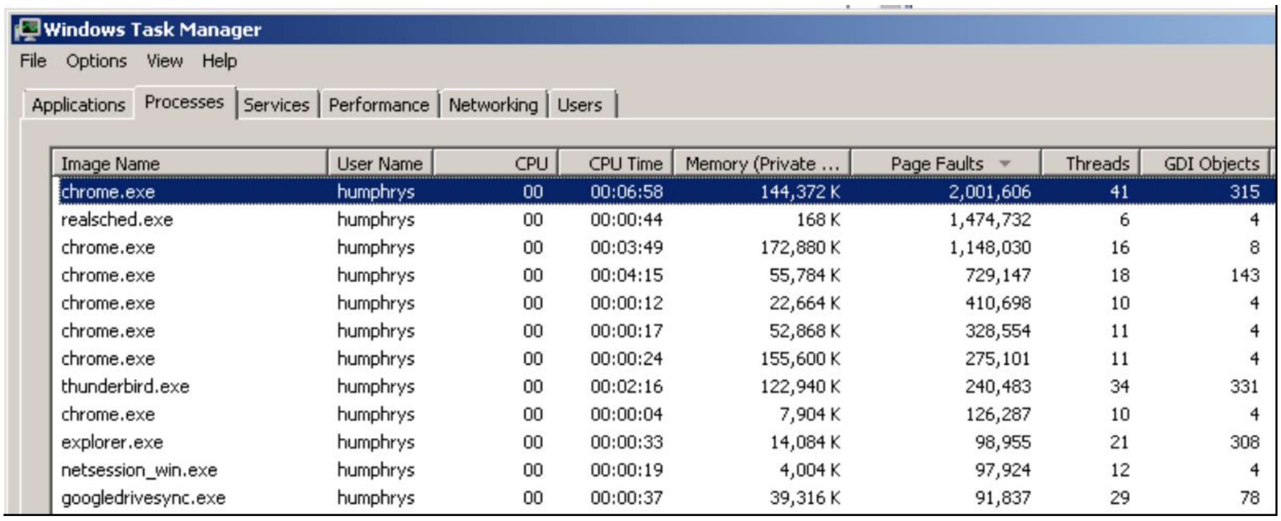
- Major (hard) page fault - Need a disk read. Takes much longer to return.

## Viewing page faults

- On Linux we can look at major and minor page faults using ps:

```
# show min_flt
# and maj_flt

$ ps -A -o user,pid,ppid,min_flt,maj_flt,comm,args

# sorted descending by number of minor page faults:

$ ps -A -o user,pid,ppid,min_flt,maj_flt,comm,args | sort -nr -k 4

USER        PID  PPID  MINFL  MAJFL COMMAND         COMMAND
root       1198     1 20509703     0 xe-daemon       /bin/bash
/usr/sbin/xe-daemon
root       2920     1 10267099     0 sshd            /usr/sbin/sshd
root       3172     1 6000382     1 xinetd          /usr/sbin/xinetd
root      11034     1 1176904     0 fail2ban-server /usr/bin/python
/usr/bin/fail2ban-server
root       5293     1 975768     28 nscd            /usr/sbin/nscd
root       3149     1 193727      0 cron            /usr/sbin/cron
```

- On Windows we can look at combined page faults using Windows Task Manager:

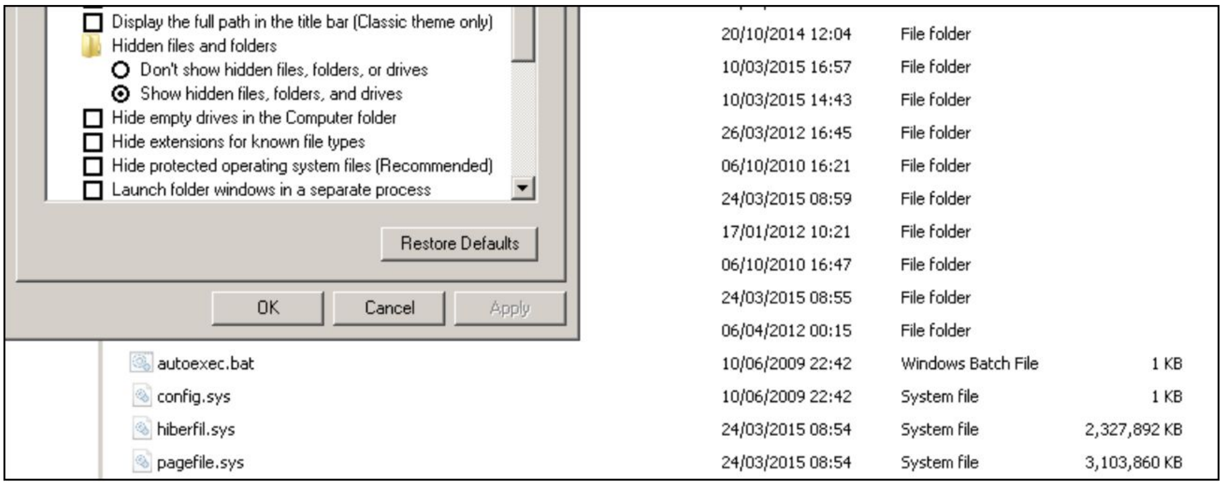

Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users

| Image Name | User Name | CPU | CPU Time | Memory (Private ... | Page Faults | Threads | GDI Objects |
|---|---|---|---|---|---|---|---|
| chrome.exe | humphrys | 00 | 00:06:58 | 144,372 K | 2,001,606 | 41 | 315 |
| realsched.exe | humphrys | 00 | 00:00:44 | 168 K | 1,474,732 | 6 | 4 |
| chrome.exe | humphrys | 00 | 00:03:49 | 172,880 K | 1,148,030 | 16 | 8 |
| chrome.exe | humphrys | 00 | 00:04:15 | 55,784 K | 729,147 | 18 | 143 |
| chrome.exe | humphrys | 00 | 00:00:12 | 22,664 K | 410,698 | 10 | 4 |
| chrome.exe | humphrys | 00 | 00:00:17 | 52,868 K | 328,554 | 11 | 4 |
| chrome.exe | humphrys | 00 | 00:00:24 | 155,600 K | 275,101 | 11 | 4 |
| thunderbird.exe | humphrys | 00 | 00:02:16 | 122,940 K | 240,483 | 34 | 331 |
| chrome.exe | humphrys | 00 | 00:00:04 | 7,904 K | 126,287 | 10 | 4 |
| explorer.exe | humphrys | 00 | 00:00:33 | 14,084 K | 98,955 | 21 | 308 |
| netsession_win.exe | humphrys | 00 | 00:00:19 | 4,004 K | 97,924 | 12 | 4 |
| googledrivesync.exe | humphrys | 00 | 00:00:37 | 39,316 K | 91,837 | 29 | 78 |

- Windows 7 Task Manager.
- The above combines both hard and soft page faults.
- "Resource Monitor" (see above) shows hard faults per second.

## Where to swap to?

- Swap to an area of disk dedicated to swapping. Maybe entirely separate to file system.
- On UNIX/Linux, swap to a partition that has no file system on it.
- User doesn't have to see it. Doesn't have file names. Not addressable by user.
- On Windows, swap to the named file `C:\pagefile.sys`
- Might move this onto its own partition.



☐ Display the full path in the title bar (Classic theme only)
Hidden files and folders
○ Don't show hidden files, folders, or drives
[✓ Show hidden files, folders, and drives]
☐ Hide empty drives in the Computer folder
☐ Hide extensions for known file types
☐ Hide protected operating system files (Recommended)
☐ Launch folder windows in a separate process

| | | | |
|---|---|---|---|
| | 20/10/2014 12:04 | File folder | |
| | 10/03/2015 16:57 | File folder | |
| | 10/03/2015 14:43 | File folder | |
| | 26/03/2012 16:45 | File folder | |
| | 06/10/2010 16:21 | File folder | |
| | 24/03/2015 08:59 | File folder | |
| | 17/01/2012 10:21 | File folder | |
| | 06/10/2010 16:47 | File folder | |
| | 24/03/2015 08:55 | File folder | |
| | 06/04/2012 00:15 | File folder | |
| autoexec.bat | 10/06/2009 22:42 | Windows Batch File | 1 KB |
| config.sys | 10/06/2009 22:42 | System file | 1 KB |
| hiberfil.sys | 24/03/2015 08:54 | System file | 2,327,892 KB |
| pagefile.sys | 24/03/2015 08:54 | System file | 3,103,860 KB |

- Windows: In Folder Options, "Show hidden files" and uncheck "Hide protected OS files".
- Then can see C:\pagefile.sys (size here 3 G).

## Hibernation

- Another form of dumping memory to disk is hibernation. Power down with programs open. Write dump of memory to disk. When power up, read image of memory back from disk.
- On Windows, memory image saved in C:\hiberfil.sys

# Files

## Files

- File Systems

## Contiguous file allocation

- Files all in one unbroken sequence on the physical disk.
- Problems like with contiguous memory allocation.
- What happens if file grows? Has to be rewritten into larger slot. Takes ages.
- Many files grow slowly,
    - e.g. text editor files.
- But some files grow quickly, unpredictably, or without limit.
    - e.g. HTTP access log, or zip or tar archive file:

```
tar -cf file.tar $home
```

    - Grows from 0 k to 10 G in a minute or two.

## Non-contiguous allocation

- Like with paging in memory, disk is divided into blocks.
- Blocks are of fixed size, say 1 k or 4 k.
- On Linux there are ways to query block size of different devices at command line.
- On DCU Linux, may not have permission. So here is a quick alternative way:

```
Create new file. Will get one block allocated for it:
$ echo 1 > newfile

See how much space allocated for it (one block).
$ du -h newfile
4.0K
```

- File is held in a collection of blocks scattered over the disk.
- If file needs more blocks, it can take them from anywhere on disk that blocks are free.

## Index of where the blocks of the file are

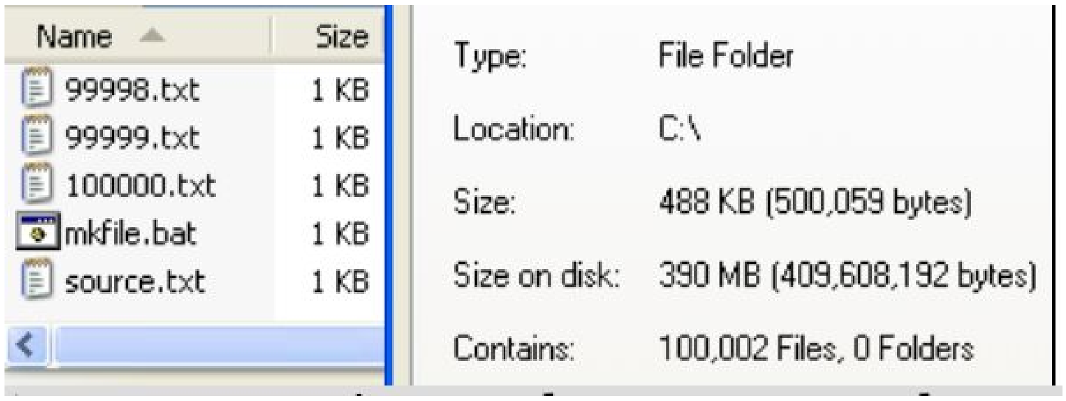
- Like pages in memory, blocks can "flow" like liquid into slots around the disk. Don't all need to be in the same place.
- Need some index of where the bits of the file are.
- Various indexing systems using linked lists or tables:
    - File Allocation Table (FAT)
    - NTFS
    - UNIX inode
- See Demonstration of fragmentation (files split into multiple parts).

## Shell script to see blocks allocated to files

```
# compare actual file size with blocks used

for i in *
do
 ls -ld $i
 du -h $i
done
```

- Results like (on system with 1 k blocks):

```
-rwxr-xr-x 1 me mygroup 857 Jun 27  2000 synch
1.0K    synch

-rwxr--r-- 1 me mygroup 1202 Oct 25  2013 profile
2.0K    profile

-rwxr-xr-x 1 me mygroup 1636 Oct 28  2009 yo.java
2.0K    yo.java

-rwxr--r-- 1 me mygroup 2089 Oct  8 00:03 flashsince
3.0K    flashsince

-rwxr-xr-x 1 me mygroup 9308 Oct 19  2010 yo.safe
10K     yo.safe
```

- 

| Name | Size |
|---|---|
| 99998.txt | 1 KB |
| 99999.txt | 1 KB |
| 100000.txt | 1 KB |
| mkfile.bat | 1 KB |
| source.txt | 1 KB |

|  |  |
|---|---|
| Type: | File Folder |
| Location: | C:\ |
| Size: | 488 KB (500,059 bytes) |
| Size on disk: | 390 MB (409,608,192 bytes) |
| Contains: | 100,002 Files, 0 Folders |

- An extreme experiment to demonstrate wasted space ("slack space") in file systems.
- This person makes 100,000 files of 5 bytes each.
- This is only 500 k of actual data.
- But it needs 400 M of disk space to store it.
- From here.

## Contiguous file allocation is good where possible

- Unlike in memory, where contiguous allocation is dead, in files it has made a bit of a comeback.
- The reason is that there is a lot of disk space and it is cheap, but the speed of read/writing disk has not increased so much - indeed it has got far worse relative to CPU speed.
- To write a contiguous file to disk you don't have to jump the disk head around the disk. You just keep writing after the last location you wrote - the head is in the correct position.
- So modern systems try to use contiguous allocation for small files, only use non-contiguous for large files. To achieve this, they may allocate more space than possibly needed. Some wasted disk space is worth it if it makes disk I/O a lot faster.
- Look up Demonstration of fragmentation (files split into multiple parts) and defragmentation (reducing such splitting and making files contiguous)

## Cache blocks in RAM for speed

- Also to speed things up: OS caches recently-accessed blocks in memory in case needed again. Avoid another disk I/O.

## RAM drive

- RAM drive - RAM formatted with a file system.
- Very fast, but volatile.
- Will be cleared after reboot. Needs to be loaded up with files from disk (non-volatile).
- Need to write changes to files to actual disk or they will be lost.
- Automatic software can take care of the above two.
- Battery-backup RAM drive - survive through power outage.

# Using Files

## Using files

- File - A named section of disk.
- Files implementation: Not necessarily a contiguous section of disk (but that fact may be hidden from users and programs).
- Normally both user and programmer never deal with disk directly, but only by calling named files.
- In some high-performance application (e.g. writing a high-speed search engine), you may need to implement your own file system, but this is obviously difficult and full of dangers.

## File Types

- List of file formats
- Alphabetical list of file extensions
- `file` - query file type
- Programs (machine readable)
  - Extensions: class, exe, com, o, obj, a, dll
  - See also Program File Types
- Program source code (human readable)
  - java, c, cxx, h, hxx, pas, asm
- Programs (human readable) - interpreted scripts
  - js, php, sh, bat, pl
- Program data (machine readable). Often strictly formatted. Precise length of each field pre-defined (for ease of machine reading, and so data can be read into pre-defined fixed-size program variables).
  - Database files.
  - Documents for display. e.g. Word docs (doc), ps, pdf, rtf, tex, dvi
  - Multimedia files - images, audio, video. - gif, jpg, jpeg, mpg, mpeg, ram, avi, qt, au.
- Program data (human readable). Often variable size, free-form text.
  - Preferences files, rc.
  - Documents for display. e.g. HTML docs (htm, html, shtml), xml, Office xml (docx), latex, txt
  - Human readable program data - xml, json
  - See also Human readable program data
- Log files.
- Archive files - tar
- Compressed files - zip, arc, gz, Z.

## File system divisions

- Windows file system can spread over multiple pieces of hardware. Each given its own (single-letter) drive:
  `drive:\dir\file`
- Can also partition a single piece of hardware into multiple drives.
- UNIX file system can spread over multiple pieces of hardware too. But everything appears as sub-directories of a single file hierarchy.
- Path may indicate hardware, something equivalent to:
  `/drive/dir/file`
- or may hide hardware entirely:
  `/dir/file`
- df
- Distributed file systems
- Network file systems

## Hierarchical file system

- Can organise files in separate dirs (Many web authors seem not to have discovered sub-dirs!).
- Crucial to keep user files separate from system files (Why?).
- Windows `C:\Users\me`
- UNIX `$HOME`
- Can reuse the same file names in different sub-dirs (like index.html).

## Long file names

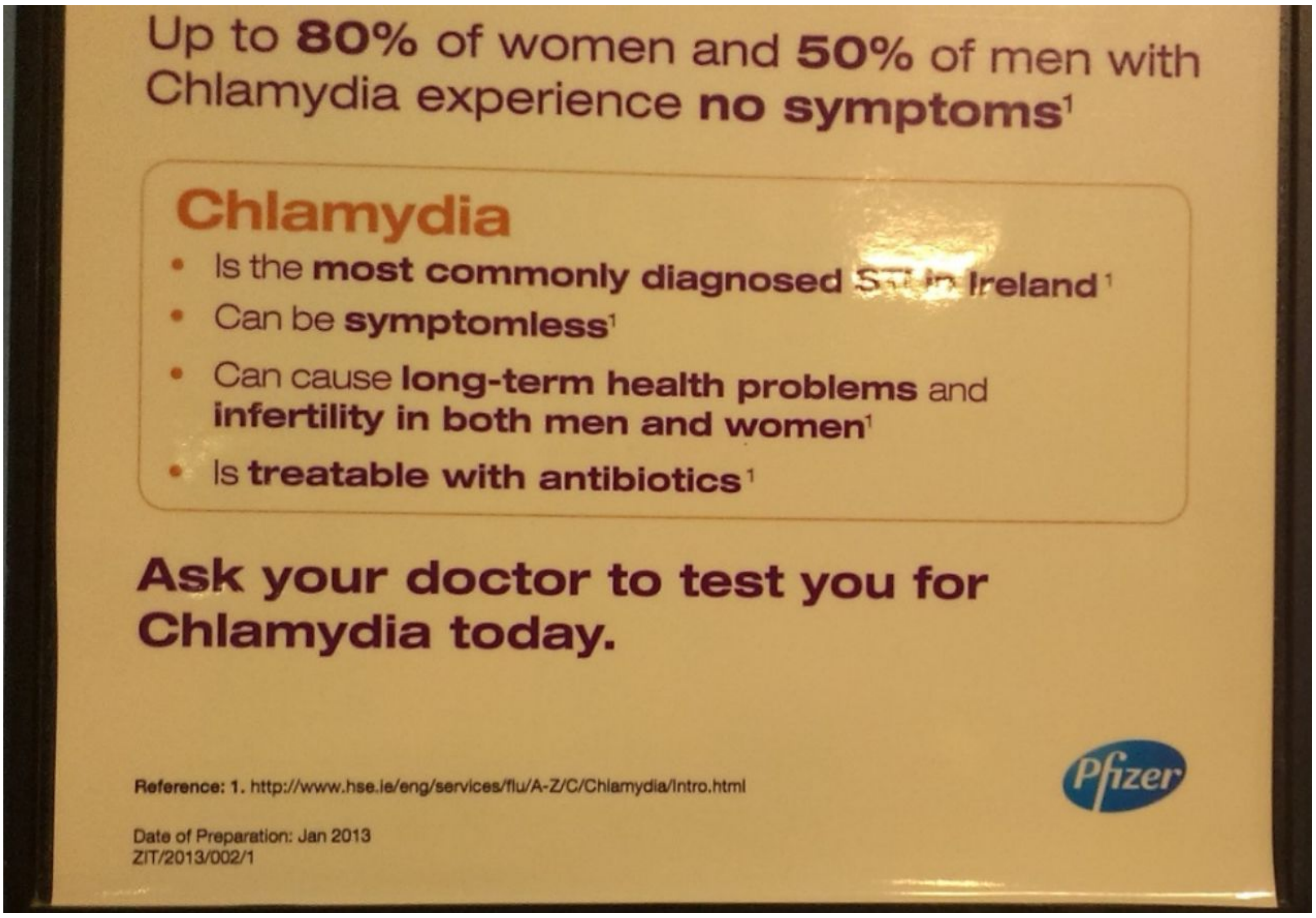
- All modern OS's allow long filenames:
  `Photos.kenya.apr.1963.html`
- Legacy systems:
  - DOS and Windows before Windows 95 had 8 char name, 3 char extension:
    `phka0463.htm`
- VM/CMS had 8 char filenames and no sub-directories!

## Short file names are good for ..

- Short file names are good, though, for:

1. File names you type. e.g. If you are typing file names at command-line. All-lower-case is easiest to type.
2. Program names at the command-line (i.e. the program you call has a short filename). sed, grep, ls, cut, etc. All-lower-case is easiest to type.
3. Some people say also URLs?

- Maybe you should never type URLs. At most you type the host name that you saw somewhere. For everything else you cut and paste, or click.
- Maybe short URLs: http://en.wikipedia.org/wiki/Othello make the web a more pleasant experience than long URLs:
    - http://dmoztools.net/Arts/Literature/World_Literature/British/Shakespeare/Works/Plays/Tragedies/Othello/.
- It is nice to have short, "guessable" URLs.
- See "URL as UI"
- See URL shortening. (Used e.g. on Twitter.)


Up to **80%** of women and **50%** of men with Chlamydia experience **no symptoms**[1]

**Chlamydia**

- Is the **most commonly diagnosed STI in Ireland**[1]
- Can be **symptomless**[1]
- Can cause **long-term health problems** and **infertility in both men and women**[1]
- Is **treatable with antibiotics**[1]

**Ask your doctor to test you for Chlamydia today.**

**Reference: 1.** http://www.hse.ie/eng/services/flu/A-Z/C/Chlamydia/intro.html

Date of Preparation: Jan 2013
ZIT/2013/002/1

Pfizer

- Short URLs should probably be used in posters and ads:
- This health poster on campus caught my eye.
- This probably should use a shorter, and lowercase, URL, like: hse.ie/chlamydia

## Symbolic link (cross-link, breaking the hierarchy, "shortcut") in UNIX

- Directory
  - Can selectively break the hierarchy with shortcuts.
    ```
     ln -s dir shortcut
    ```
  - or in Windows see "Create Shortcut"
  - e.g. on one system I used:
    ```
    $ ls -l /bin
    lrwxrwxrwx   1 root     root           9 Apr 14  1997 /bin ->
    ./usr/bin
    ```
- File
  - Can also just give a file multiple names:
    ```
     ln -s file secondname
    ```
  - e.g. on DCU Linux:
    ```
    $ ls -l /bin/ls
    lrwxrwxrwx 1 root root 11 Apr  8 21:49 /bin/ls -> /usr/bin/ls

    $ ls -l /usr/bin/ps
    lrwxrwxrwx 1 root root 7 Feb 12 12:14 /usr/bin/ps -> /bin/ps
    ```
- Q. Why do programs sometimes call a specific path to a program, e.g. they call /bin/ls rather than just ls ?
  - Can do this on Windows as well (have multiple shortcuts to a data file or program).

## Problems with cross-links

- With shortcuts, if doing a recursive search of disk, can get infinite loop problems, or at least duplication. e.g. List all files on disk. If follow symbolic links may list files twice.
- Q. Also, if delete file, do you delete symbolic link? If so, how do you find them - do you have reverse directory of them? Also, I make symbolic link to other user's file. They delete file. They can't delete my link.
- A. If link doesn't work, so what. Might even leave it dangling as reminder.
- Security
  - If your directory is accessible by others on your local machine, someone on your machine can make it readable by the world on the Web (either maliciously or accidentally):
    ```
    cd     /homes/your-userid/public_html
    ln -s  /homes/other-userid/dir          shortcut
    ```
  - The world can then read other user's directory through:
    ```
    http://host/~your-userid/shortcut/
    ```
  - Has valid uses too. Might want to make one of your own dirs visible without having to have it under public_html, e.g. public_html disk is full, dir is on another disk.
  - Another example - SAMBA or read-write ftp may only drop you in home directory rather than root directory and you may not be able to go upwards. What you do is put symbolic links in your home directory and you can access any directory through them:
    ```
    ln -s /var/mail  email
    ln -s /htdocs    ht
    ```

## "Hierarchy with some cross-links" a very powerful model

- General conclusion is that a basic hierarchy, with some cross-links for difficult points, is excellent way to structure complex data (e.g. Open Directory) - rather than total cross-link free-for-all on one hand (e.g. the Web with just search engines and no directories), or rigid hierarchy on other (e.g. Dewey library system).
- Interestingly, family trees are also basically hierarchical, with arbitrary cross-links, rather than strictly hierarchical as many people seem to think.

## Recycle bin (Windows)

- Windows Recycle bin visible through GUI, but also visible as directory through Windows command line:
  - `cd c:\$recycle.bin`
  - `dir` (apparently empty)
  - `dir /ah` (show hidden files)

## Backup

- If it's data (1's and 0's), there's no real excuse for losing it. You can make automated copies and store them all over the world. Disk space is big and cheap. Machines are often idle. The network is always on. Backups can be automated across the network by scripts.
- In future, backup and long-term storage will be increasingly important service, like a bank.
- Removable media - DVDs, CDs, tapes, USB keys, external hard disk.
- v.
- Backup to cloud / server. Distributed file system. Network read-write ftp, automated scripts, mirrors.

- List of backup software
- Comparison of backup software
- Online backup services
- Comparison of online backup services

## Other people back you up

- Even if you back up nothing, your web pages are being backed up by other people:
- Google cache (click on "Cached")
- Microsoft cache (click on "Cached page")
- Internet Archive
- Flickr - network photo storage
- Other social media.
- Often our data is on a remote backed-up server by default

## Backup policy

1. Periodically dump entire file system to backup.
   v.
2. Keep a running "mirror", and only backup things that have changed since last time they were synch-ed.

- Perhaps only backup user files.
- OS, system and application files can be recovered from install CDs / tapes.
- Which of these is the most dangerous:
    1. Keep 1 synchronised copy of your files. Backup the changes every night.
    2. Keep 1 synchronised copy of your files. Backup the changes every hour.
    3. Take a copy of all of your files once a week. Keep all these old copies. Do no backups at all during the week.
    4. Take a copy of all of your files once a month. Keep all these old copies. Do no backups at all during the month.
- Remember - it may take days or even months before an intrusion and destruction, or accidental damage, is noticed.
- User may realise 2 years later that he has deleted some file and needs it back.
- VAX/VMS (DEC) could be set to keep all drafts of a file since created.
- The equivalent of `ls` would hide all except the latest one by default. Unless explicitly asked otherwise.
- Programming with DCL would work with the latest one by default. Unless explicitly asked otherwise.
- Lot to be said for such an approach, now that disk space is cheap.
- Versioning file systems
- Google Docs saves all old drafts/versions of docs.