Virtual Memory Wrap-up;
File System Interface
Memory-mapped Files

- Memory mapping a file can be accomplished by mapping a disk block to one or more pages in memory.

- A page-sized portion of the file is read from the file system into a physical page. Subsequent read() and write() operations are handled as memory (not disk) accesses.

- Writing to the file in memory is not necessarily synchronous to the file on disk. The file can be committed back to disk when it's closed.
Memory-mapped Files

process A virtual memory

process B virtual memory

disk file
Prepaging

- **Prepaging**: In order to avoid the initial number of page faults, the system can bring into memory all the pages that will be needed all at once.

- This can also be applied when a swapped-out process is restarted. The smart thing to do is to remember the working set of the process.

- One question that arises is whether all the pages brought in will actually be used…

- Is the cost of prepaging less than the cost of servicing each individual page fault?
File System Topics

- File Concept
- Access Methods
- Directory Structure
- File System Mounting
- File Sharing
- Protection
File Concept

- A file is a named collection of related information recorded on secondary storage.
- “Contiguous” logical address space.
- File types:
  - Data:
    - numeric.
    - character.
    - binary.
  - Program (executable).
File Structure

- None: just a sequence of words or bytes.
- Simple **record** structure:
  - Lines,
  - Fixed length,
  - Variable length.
- Complex Structures:
  - Formatted document,
  - Relocatable load file.
- Can simulate last two with first method by inserting appropriate control characters.
- Who decides:
  - Operating system,
  - Program.
File Attributes

- **Name** – only information kept in human-readable form.
- **Type** – needed for systems that support different types.
- **Location** – pointer to file location on device.
- **Size** – current file size.
- **Protection** – controls who can do reading, writing, executing.
- **Time, date, and user identification** – data for protection, security, and usage monitoring.

Information about files is kept in the directory structure, which is maintained on the disk.
File Operations

- Create.
- Write.
- Read.
- Seek.
- Delete.
- Truncate (reset size to 0, keep current attributes).
- Open($F_i$) – search the directory structure on disk for entry $F_i$, and move the content of entry to memory.
- Close ($F_i$) – move the content of entry $F_i$ in memory to directory structure on disk.
## File Types: Name and Extension

<table>
<thead>
<tr>
<th>file type</th>
<th>usual extension</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>executable</td>
<td>exe, com, bin,</td>
<td>read to run machine-language program</td>
</tr>
<tr>
<td></td>
<td>or none</td>
<td></td>
</tr>
<tr>
<td>object</td>
<td>obj.o</td>
<td>compiled, machine language, not linked</td>
</tr>
<tr>
<td>source code</td>
<td>c, cc, java, pas,</td>
<td>source code in various languages</td>
</tr>
<tr>
<td></td>
<td>asm, a</td>
<td></td>
</tr>
<tr>
<td>batch</td>
<td>bat, sh</td>
<td>commands to the command interpreter</td>
</tr>
<tr>
<td>text</td>
<td>txt, doc</td>
<td>textual data, documents</td>
</tr>
<tr>
<td>word processor</td>
<td>wp, tex, rtf,</td>
<td>various word-processor formats</td>
</tr>
<tr>
<td></td>
<td>doc</td>
<td></td>
</tr>
<tr>
<td>library</td>
<td>lib, a, so, dll,</td>
<td>libraries of routines for programmers</td>
</tr>
<tr>
<td></td>
<td>mpeg, mov, rm</td>
<td></td>
</tr>
<tr>
<td>print or view</td>
<td>arc, zip, tar</td>
<td>ASCII or binary file in a format for printing or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>viewing</td>
</tr>
<tr>
<td>archive</td>
<td>arc, zip, tar</td>
<td>related files grouped into one file, sometimes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>compressed, for archiving or storage</td>
</tr>
<tr>
<td>multimedia</td>
<td>mpeg, mov, rm</td>
<td>binary file containing audio or A/V information</td>
</tr>
</tbody>
</table>
Access Methods

- **Sequential Access**
  - read next
  - write next
  - reset
  - no read after last write (rewrite)

- **Direct Access**
  - read \( n \)
  - write \( n \)
  - position to \( n \)
  - read next
  - write next
  - rewrite \( n \)

\( n = \) relative block number
Sequential-access File

beginning  current position  end

rewind  read or write

04/05/2010  CSCI 315 Operating Systems Design
Simulation of Sequential Access on a Direct-access File

<table>
<thead>
<tr>
<th>sequential access</th>
<th>implementation for direct access</th>
</tr>
</thead>
<tbody>
<tr>
<td>reset</td>
<td>( cp = 0; )</td>
</tr>
<tr>
<td>read next</td>
<td>( \text{read } cp; ) &lt;br&gt; ( cp = cp+1; )</td>
</tr>
<tr>
<td>write next</td>
<td>( \text{write } cp; ) &lt;br&gt; ( cp = cp+1; )</td>
</tr>
</tbody>
</table>
Example of Index and Relative Files

<table>
<thead>
<tr>
<th>last name</th>
<th>logical record number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adams</td>
<td></td>
</tr>
<tr>
<td>Arthur</td>
<td></td>
</tr>
<tr>
<td>Asher</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Smith</td>
<td></td>
</tr>
</tbody>
</table>

Index file

Relative file

Smith, John | social-security | age
Directory Structure

**Directory**: a symbol table that translates file names into directory entries.

Both the directory structure and the files reside on disk. Backups of these two structures are kept on tapes.
Partitions and Directories
(File system organization)
Operations on Directories

- Search for a file.
- Create a file.
- Delete a file.
- List a directory.
- Rename a file.
- Traverse the file system.
Goals of Directory Logical Organization

- **Efficiency** – locating a file quickly.
- **Naming** – convenient to users.
  - Two users can have same name for different files.
  - The same file can have several different names.
- **Grouping** – logical grouping of files by properties, (e.g., all Java programs, all games, …)
Single-Level Directory

A single directory for all users.

**Drawbacks:**
- Naming problem
- Grouping problem
Two-Level Directory

A separate directory for each user.

- Path name.
- Can have the same file name for different users.
- Efficient searching.
- No grouping capability.
Tree-Structured Directories

Diagram of a tree-structured directory system with nodes labeled as follows:
- root
- spell
- bin
- programs
- stat
- mail
- dist
- find
- count
- hex
- reorder
- p
- e
- mail
- prog
- copy
- prf
- exp
- reorder
- list
- find
- hex
- count
- list
- obj
- spell
- all
- list
- first
Tree-Structured Directories (Cont.)

- Efficient searching.
- Grouping Capability.
- Current directory (working directory):
  - `cd /spell/mail/prog`,
  - `type list`. 
Tree-Structured Directories (Cont.)

- **Absolute** or **relative** path name.
- Creating a new file is done in current directory by default.
- Delete a file
  
  `rm <file-name>`

- Creating a new subdirectory is done in current directory.
  
  `mkdir <dir-name>`

  Example: if in current directory `/mail`

  `mkdir count`

```
  mail
    
    prog copy prt exp count
```

Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”.

04/05/2010  CSCI 315 Operating Systems Design  23
Acyclic-Graph Directories

Have shared subdirectories and files.
Acyclic-Graph Directories (Cont.)

- Two different names (aliasing).
- If dict deletes list ⇒ dangling pointer.

Solutions:
- Backpointers, so we can delete all pointers. Variable size records a problem.
- Backpointers using a daisy chain organization.
- Entry-hold-count solution.
General Graph Directory

![Diagram of a general graph directory]
General Graph Directory (Cont.)

• How do we guarantee no cycles?
  – Allow only links to file not subdirectories.
  – Garbage collection.
  – Every time a new link is added use a cycle detection algorithm to determine whether it is OK.
File System Mounting

- A file system (partition) must be **mounted** before it can be accessed.
- A unmounted file system needs to be attached to a **mount point** before it can be accessed.
File Sharing

- Sharing of files on multi-user systems is desirable.
- Sharing may be done through a protection scheme.
- On distributed systems, files may be shared across a network.
- Network File System (NFS) is a common distributed file-sharing method.
Protection

- File owner/creator should be able to control:
  - what can be done,
  - by whom.

- Types of access:
  - Read,
  - Write,
  - Execute,
  - Append,
  - Delete,
  - List.
Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users
  
  a) owner access
  
  7 ⇒ 1 1 1
  
  RWX

  b) group access
  
  6 ⇒ 1 1 0
  
  RWX

  c) public access
  
  1 ⇒ 0 0 1

- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say game) or subdirectory, define an appropriate access.

  owner  
  
  group  
  
  public  
  
  chmod 781 game

Associate a group with a file: chgrp G game