Linux Case Study

OPS Lecture 18, G53OPS/G52OSC

Geert De Maere
(Jason Atkin – OSC)
Geert.DeMaere@Nottingham.ac.uk

University Of Nottingham
United Kingdom

2015

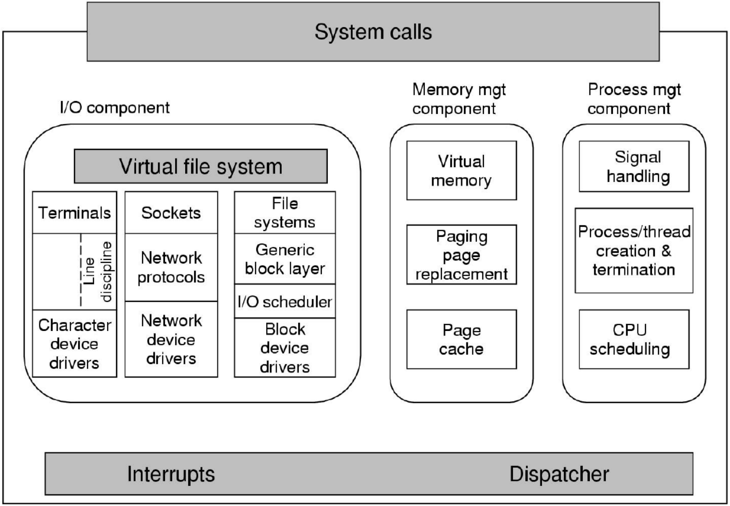

Figure: Structure of the Linux Kernel (Tanenbaum)

- Linux uses a **pure demand paging** approach (i.e. no pre-paging or working sets) and pages are reclaimed by the **paging daemon**
  - Loading a process **sets up the page table** but does not load the pages
- "**Swapping**" is implemented on a **page granularity** ($\Leftrightarrow$ entire process)
  - I.e. paging and virtual memory are used
  - The process does not have to be entirely in memory

- Linux's **page replacement algorithm** is a modified version of the **second chance clock algorithm** based on **age**:
  - Each pass of the clock reduces the **age value**
  - Each **reference** to a page **increases the age value**
- The age value is used as a basis for the **least recently used** algorithm

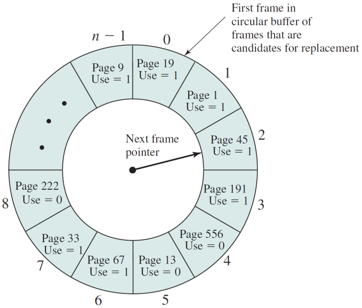

Figure: Memory Image (Tanenbaum)

- Pages can be mapped:
  - Onto files on the disk, e.g. for **text segments** and **memory mapped files**
  - Onto a **paging file** or **paging partition**
- **Partitions are faster** since it avoids file system overheads, pages are usually written contiguously
- A **next fit algorithm** is used for swap partitions, aiming to write pages **contiguously**

- **Minix file system**: the maximum file size was 64MB and file names were limited to 14 characters
- The “**extended file system**” (extfs): file names were 255 characters and the maximum file size was 2 GB
- The “**ext2**” file system: larger files, larger file names, better performance
- The “**ext3-4**” file system: journaling etc.

- Anything capable of handling input or output of data is regarded as a file in Linux
  - **Character special files** model serial devices (e.g. keyboards, mice, printers)
  - **Block special files** model raw block devices (e.g. hard drives without file systems, used for swap partitions)
- **Directories** are **implemented as files** and lookups take place in a similar way to Unix

- **Files** and **directories** are manipulated through **system calls**, e.g. `open` and `close`
- **File descriptors** are non-negative integers used as an index for the **file descriptor table** (0, 1, and 2 are used for standard input, output, and error)

System Call	Description
`fd = creat(name, mode)`	Create a new file
`fd = open(file, how, ...)`	Open file
`s = close(fd)`	Close file
`n = read(fd, buffer, nbytes)`	Read data into buffer
`n = write(fd,buffer, nbytes)`	Write data into buffer
`position = lseek(fd, offset, whence)`	Move file pointer

Table: Examples of system calls for file manipulation

System Call	Description
`s = mkdir(path, mode)`	Create a new directory
`s = rmdir(path)`	Remove directory
`s = link(oldpath, newpath`	Create link
`n = unlink(path)`	Unlink a file
`n = chdir(path)`	Change CWD
`dir = opendir(path)`	Open directory
`s = closedir(dir)`	Close directory
`dirent = readdir(dir)`	Read one directory entry

Table: Examples of system calls for directory manipulation

- Linux and Unix use a **virtual file system** (VFS) that hides the implementation specific details behind a generic interface
- **Key objects** in the VFS implementation include:
  - **Super block object** containing file system information for every mounted file system (e.g. number of i-nodes and access to i-nodes)
  - **File objects** for in memory representations of files created by an `open()` call , and containing e.g a file pointer, file permissions, etc.
  - **I-node object** containing information on individual files
  - **Dentry objects** representing directory entries and which are cached
- Objects contain **pointers to function tables** that list addresses for the actual **implementations**

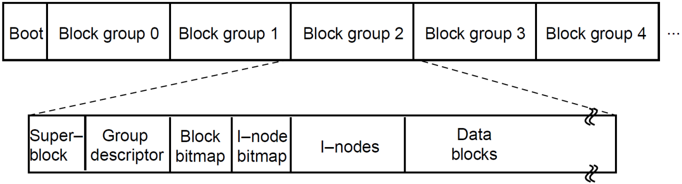

Figure: Ext2 Partition Layout (Tanenbaum)

- The **superblock** contains file system information (e.g. the number of i-nodes, disk blocks)
- The **group descriptor** contains bitmap locations, the number of free blocks, i-nodes and directories
- A **data block bitmap** and **i-node bitmap**, used to keep track of free disk blocks and i-nodes (Unix uses lists)
- A **table of i-nodes** containing file and disk block information
- **Data blocks** containing file and directory blocks

- An ext2 partition is split into several **block groups** to:
  - **Reduce fragmentation** by storing i-nodes and files, and parent directories and files in the same block group if possible
  - Reduce **seek times** and improve performance
- All block groups have the same size and are stored sequentially (which allows direct indexing)

- Every **directory** contains the following fields:
  - i-node number
  - Entry size in bytes
  - Type field, i.e. file, directory, special file, etc.
  - File name length in bytes
- Directories are **searched linearly** (i.e. they are unsorted) and a cache is maintained for recently accessed items

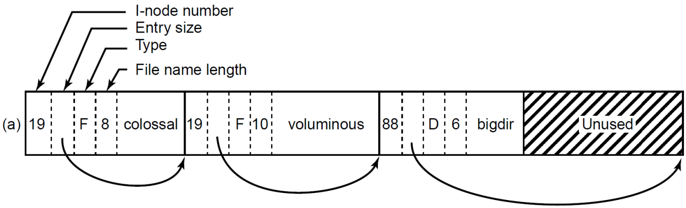

Figure: Ext2 Directory Entry (Tanenbaum)

File System

Directory Entries

- **File lookups** are similar to the Unix file system
- The **i-node structure** is similar to the Unix i-nodes
  - 12 block addresses are contained in the i-node
  - Single, double and triple indirect blocks are used

- When making changes to an Ext2 file system, files are ideally **written immediately** to prevent inconsistency:
  - This generates **significant head movement**
  - Ext2 File system is more suitable for **flash disks** (no journal)
- Ext3 builds upon the Ext2 file system by adding:
  - **Tree based structures** for directory files to facilitate indexing (HTrees)
  - **Journaling** capabilities
  - **Meta structures** remain stored in fixed locations

- Demand paging, swapping, page replacement
- File system management, including Ext2 and Ext3