*CS1101: Lecture 36*

# The OSM Level: Virtual I/O Instructions

Dr. Barry O'Sullivan
b.osullivan@cs.ucc.ie


Course Homepage
http://www.cs.ucc.ie/~osullb/cs1101



## Lecture Outline

- Introduction
- Files
- Reading a File
- Implementation of Virtual I/O
- Disk allocation strategies
- Directory Management Instructions
- Organizing On-line Files
- File Protection
- Use of Directories
- **Reading**: Tanenbaum, Chapter 6, Section 2.





## Introduction

- The OSM level instruction set contains most of the ISA level instructions, with a few new, but important, instructions added and a few potentially damaging instructions removed.
- Input/output is one of the areas where the two levels differ considerably.
- The reasons for this difference is simple: security, confidentiality, integrity.
- Second, normal, sane programmers do not want to do I/0 at the ISA level themselves because doing so is extremely tedious and complex.





## Files

- One way of organizing the virtual I/0 is to use an abstraction called a **file**.
- A file consists of a sequence of bytes written to an I/0 device.
- If the I/O device is a **storage device**, such as a disk, the file can be read back later.
- If the device is not a storage device (e.g., a printer), it cannot be read back.
- The abstraction of a file allows virtual I/0 to be organized in a simple way.

## Files – An OSM Perspective

- To the operating system, a file is normally just a sequence of bytes.
- Any further structure is up to the application programs.
- File I/O is done by system calls for opening, reading, writing, and closing files.
- Before reading a file, it must be opened (and located).
- Once a file has been opened, it can be read.



## Reading a File

- The read system call must have the following parameters, at a minimum:
  - An indication of which open file is to be read.
  - A pointer to a buffer in memory in which to put the data.
  - The number of bytes to be read.
- The read call puts the requested data in the buffer.
- Associated with each open file is a pointer telling which byte next.
- After a read it is advanced by the number of bytes read, so reads read consecutive blocks of data from the file.
- Usually, there is this pointer to a specific value, so programs can randomly access an file.





## Implementation of Virtual I/O

- We need to examine how files are organized and stored.
- A basic issue that must be dealt with by all file systems is allocation of storage.
- The **allocation unit** can be a **single disk sector**, but often it consists of a **block of consecutive sectors**.
- Another fundamental property of a file system implementation is whether a file is stored in consecutive allocation units or not.





## Disk allocation strategies

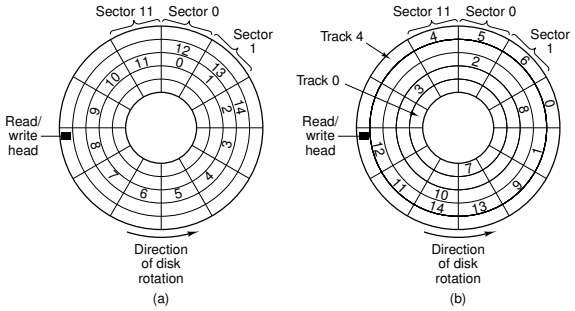


**Figure 6-20.** Disk allocation strategies. (a) A file in consecutive sectors. (b) A file not in consecutive sectors.

## Directory Management Instructions

- In the early days of computing, people kept their programs and data on punched cards in file cabinets in their offices.
- Eventually led to the idea of using the computer's secondary memory (e.g., disk) as a storage place for programs and data.
- Information that is directly accessible to the computer without the need for human intervention is said to be **on-line**.
- On-line information is stored in files, making it accessible to programs via the file I/0 instructions discussed above.
- However, additional instructions are needed to keep track of the information stored on-line, collect it into convenient units, and to protect it from unauthorized use.



## Organizing On-line Files

- The usual way for an operating system to organize on-line files is to group them into directories.
- System calls are provided for at least the following functions:
  - Create a file and enter it in a directory.
  - Delete a file from a directory.
  - Rename a file.
  - Change the protection status of a file.





## Organizing On-line Files

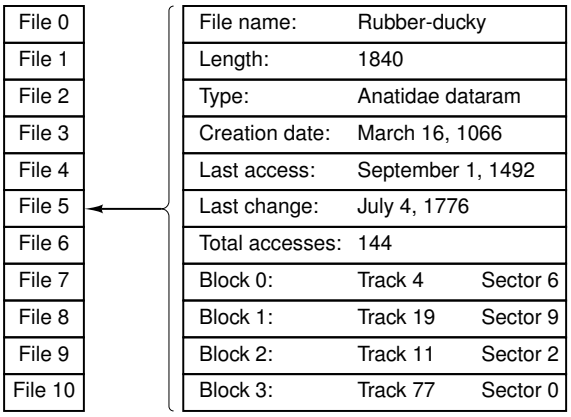


**Figure 6-22.** (a) A user file directory. (b) The contents of a typical entry in a file directory.





## File Protection

- Various protection schemes are in use.
- One possibility is for the owner of each file to specify a secret password for each file.
- When attempting to access a file, a program must supply the password, which the operating system then cheeks to see if it is correct before permitting the access.
- Another protection method is for the owner of each file to provide an explicit list of people whose programs may access that file.

## Use of Directories

- All modern operating systems allow users to maintain more than one file directory.
- Each directory is typically itself a file and, as such, may be listed in another directory, thus giving rise to a tree of directories.
- Multiple directories are particularly useful for programmers working on several projects.
- They can then group all the files related to one project together in one directory.
- While working on that project, they will not be distracted by unrelated files.
- Directories are also a convenient way for people to share files with members of their group.