File I/O

- File input/output
- Iterate through a file using `for`
- File methods read(), readline(), readlines()
- Writing to a file

What is a file?

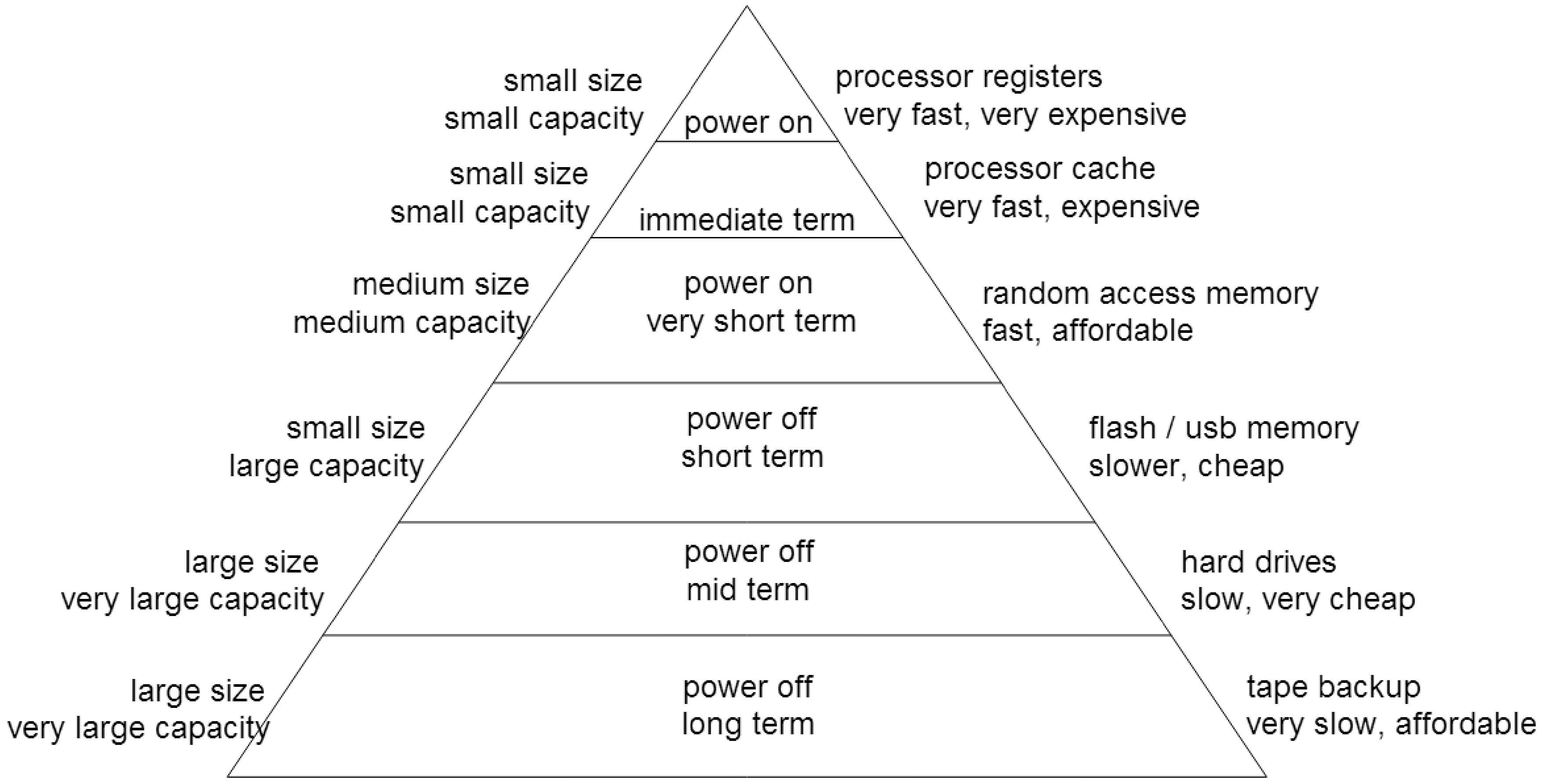
- A file is a collection of data
- Files are usually stored in non-volatile memory (that is, they don’t disappear when the power is turned off)
- A common type of file is a text file
  - A text file is a sequence of lines, e.g.
    - Computer source code
    - An HTML document (web page)
- A binary file may contain data in some non-text format
  - Music, images and video are common
  - Instrument readings

Why do we use files?

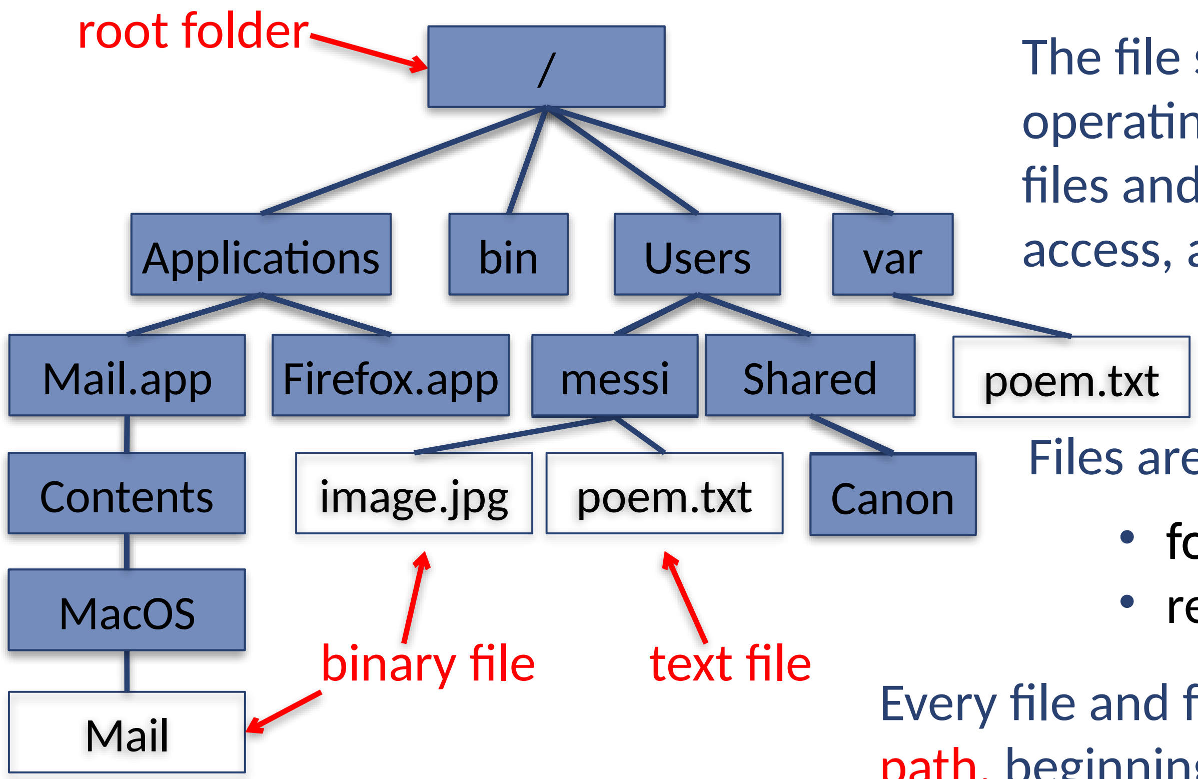
- Data may be stored in a file because
  - the data is bigger than available RAM
  - the data requires long-term storage
  - a file can be easily duplicated
  - a file can be easily transported
  - file storage is cheap
- Files are usually stored in devices at the bottom of a memory hierarchy
  - High end: fast, small, expensive, requires power (volatile)
  - Low end: big, slow, cheap, stable (non-volatile)

Computer Memory Hierarchy

by Dan Lash (.com)



Files and the file system



The file system is the part of the operating system (OS) that organizes files and provides a way to create, access, and modify files

Files are organized into a tree structure

- folders (or directories)
- regular files

Every file and folder has a name. The entire path, beginning at the root, identifies the file.

Absolute path (from root)

- `/var/poem.txt`
- `/Users/messi/poem.txt`
- `/Applications/Mail.app/`

Relative (to current working directory) path

- `messi/poem.txt`
- `messi/image.jpg`
- `Shared`

How to use a file

1. **Open the file**
2. **Read from or write to the file**
3. **Close the file**

Use mode `'r'` to open a file for reading, use mode `'w'` to open a file for writing

Use the built-in function **open(file_name, mode)** to open a file

- The first argument is a string that is the name of a file
- The second (optional) argument is the mode

```
>>> inFile = open('thisLandIsYourLand.txt', 'r')
```

open() returns an object of type "file". A file object supports several methods, including `close()`

Open file mode

The file mode defines how the file will be accessed

Mode	Description
`r`	Reading (default)
`w`	Writing (if file exists, content is wiped)
`a`	Append (if file exists, writes are appended)
`r+`	Reading and Writing
`t`	Text (default)
`b`	Binary

These are all equivalent →

```
>>> infile = open('example.txt', 'rt')
>>> infile = open('example.txt', 'r')
>>> infile = open('example.txt', 't')
>>> infile = open('example.txt')
```

Write lines to a file using `for`

- A text file is a sequence of lines delimited by '\n' (endline)
  - create (**open**) a file 'humpty.txt'
  - **write** four lines to the file
  - **close** the file

```
>>> hdList = ['Humpty Dumpty sat on a wall.', 'Humpty Dumpty had
a great fall.', "All the king's horses and all the king's men",
"Couldn't put Humpty together again!"]
>>> humptyFile = open('humpty.txt', 'w')
>>> for line in hdList:
        humptyFile.write(line + '\n')
29
32
45
36
>>> humptyFile.close()
```

- Anything you write to a text file must be a **string**
- The write method does not automatically append an **endline** – you must explicitly include a '\n' to end a line
- The write method **returns** the number of characters written

Iterate through an existing text file using `for`

- A text file is a sequence of lines delimited by '\n'.
  - **`open`** a file to create a file object for reading
  - use a **`for`** loop to iterate through a file a line at a time
  - use **`for`** the same way you would any other sequence object
  - **`close`** the file

```
>>> humptyFile = open('humpty.txt', 'r')
>>> for line in humptyFile:
         if 'Humpty' in line:
                 print(line, end = '')

Humpty Dumpty sat on a wall.
Humpty Dumpty had a great fall.
Couldn't put Humpty together again!
>>> humptyFile.close()
```

Note:

- the '\n' is included in the line that is read
- See what happens when you do not set the argument end to ''

File methods

There are several Python methods for reading from and writing to files

- **`read(n)`** returns n characters, or the entire file if parameter n is omitted
- **`readline()`** returns one line (including the endline character)
- **`readlines()`** returns the entire file as a list of strings (one line = one string)
- **`write()`** returns the number of characters written

Usage	Description
`infile.read(n)`	Read `n` characters starting from cursor; if fewer than `n` characters remain, read until the end of file
`infile.read()`	Read starting from cursor up to the end of the file
`infile.readline()`	Read starting from cursor up to and including the endline character
`infile.readlines()`	Read starting from cursor up to the end of the file and return a list of lines
`outfile.write(s)`	Write string `s` to `outfile` starting from cursor
`infile.close()`	Close `infile`

Reading a file

Create this file, named 'example.txt' using the Idle editor

```
The 3 lines in this file end with the new line character.\n
\n
There is a blank line above this line.\n
```

When the file is opened, a cursor is associated with the opened file

The initial position of the cursor is:

- at the beginning of the file, if file mode is `r`
- at the end of the file, if file mode is `a` or `w`

```
>>> infile = open('example.txt')
>>> infile.read(1)
'T'
>>> infile.read(5)
'he 3 '
>>> infile.readline()
'lines in this file end with the new line
character.\n'
>>> infile.read()
'\nThere is a blank line above this line.\n'
>>> infile.close()
>>>
```

Patterns for reading a text file

Common patterns for reading a file:

1. Read the file content into a string
2. Read the file content into a list of words
3. Read the file content into a list of lines

```
def numWords(filename):
    '''return the number of
     words in filename'''
    infile = open(filename)
    content = infile.read()
    infile.close()
    wordList = content.split()
    return len(wordList)
```

```
def numLines(filename):
    '''return the number of
    lines in filename'''
    infile = open(filename, 'r’)
    lineList = infile.readlines()
    infile.close()
    return len(lineList)
```

```
def numChars(filename):
    '''return the number of characters in filename'''
    infile = open(filename, 'r')
    content = infile.read()
    infile.close()
    return len(content)
```

Writing to a text file

```
This is the first line. Still the first line…\n
Now we are in the second line.\n
Non string value like 5 must be converted first.\n
Non string value like 5 must be converted first.\n
```

```
>>> outfile = open('test.txt', 'w')
>>> outfile.write('T')
1
>>> outfile.write('his is the first line.')
22
>>> outfile.write(' Still the first line...\n')
25
>>> outfile.write('Now we are in the second line.\n')
31
>>> outfile.write('Non string value like ' + str(5) + ' must be
converted first.\n')
49
>>> outfile.write('Non string value like {} must be converted first.\
n'.format(5))
49
>>> outfile.close()
```