Engineering 1D04

Teaching Session 8

File I/O

- Remember where we left off:
- We read all the grades from a file and then processed them.
- We then wrote them out to a text file that we could read with Microsoft Excel.

File I/O

- It's inconvenient that the files need to be in the current working directory.
- In order to remove the dependency on the working directory, filenames can also be fully qualified filenames (or just *full pathnames*).
- Lets look at the file structure of Windows to discuss the concept of full filenames.

Memory

- There are two types of memory in a computer.
  - Primary Memory
  - Secondary Memory

volatile memory

persistent memory

- Common types of primary memory are:
  - Cache
  - RAM

Memory

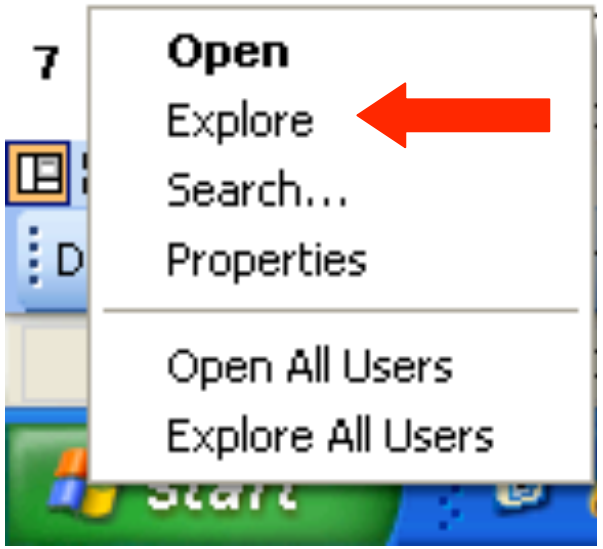
- Secondary memory devices could be:
  - DVD
  - Hard Drive
  - Magnetic Tape
  - Flash Disk
  - Floppy Drive

Memory

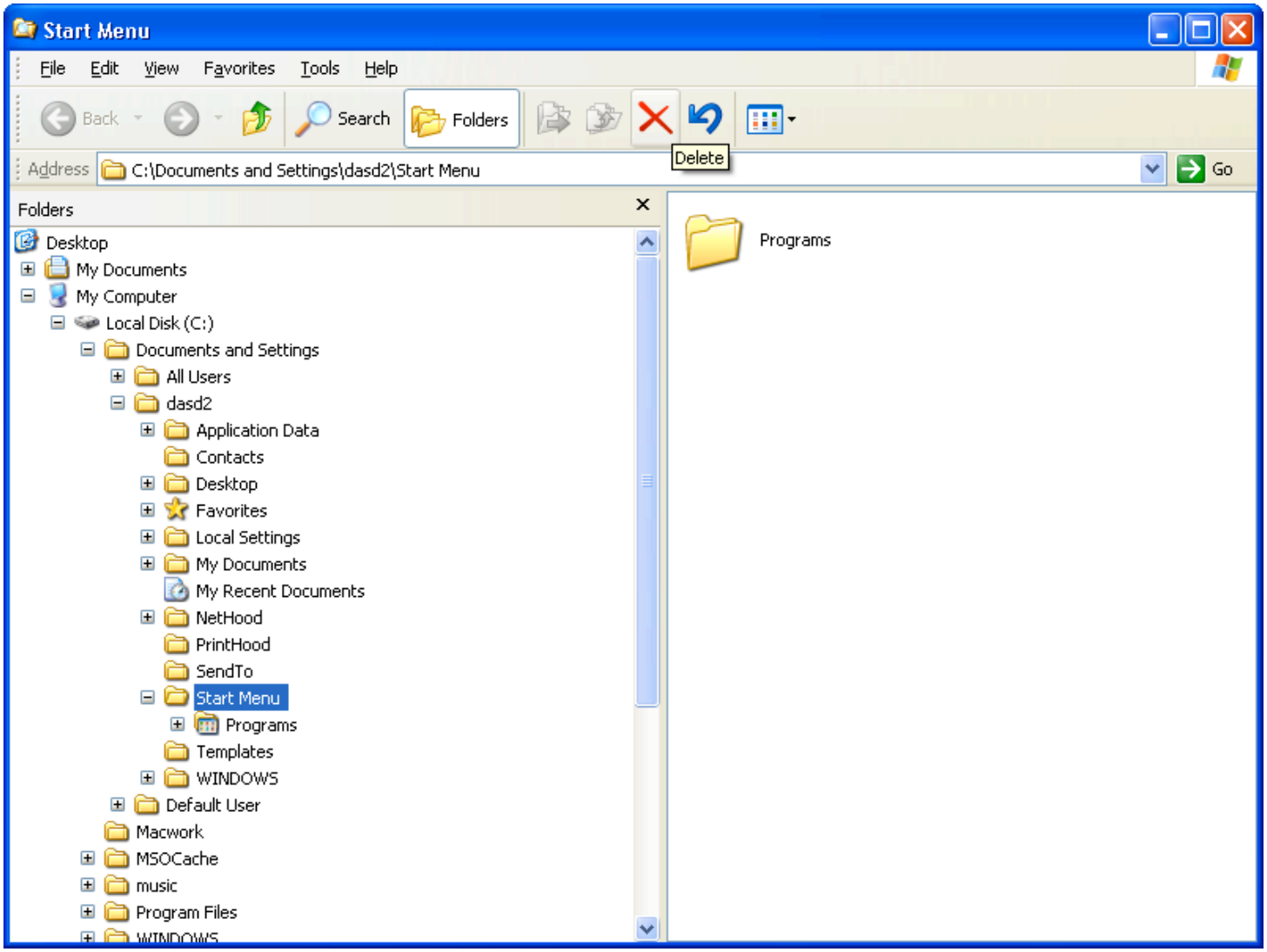
- We're particularly interested in secondary memory at this point.
- Only this type of memory can store files and folders.

Windows Explorer

- There are a number of visual ways to browse secondary memory.
- Windows Explorer:
  - Right Click on Start
  - Click Explore

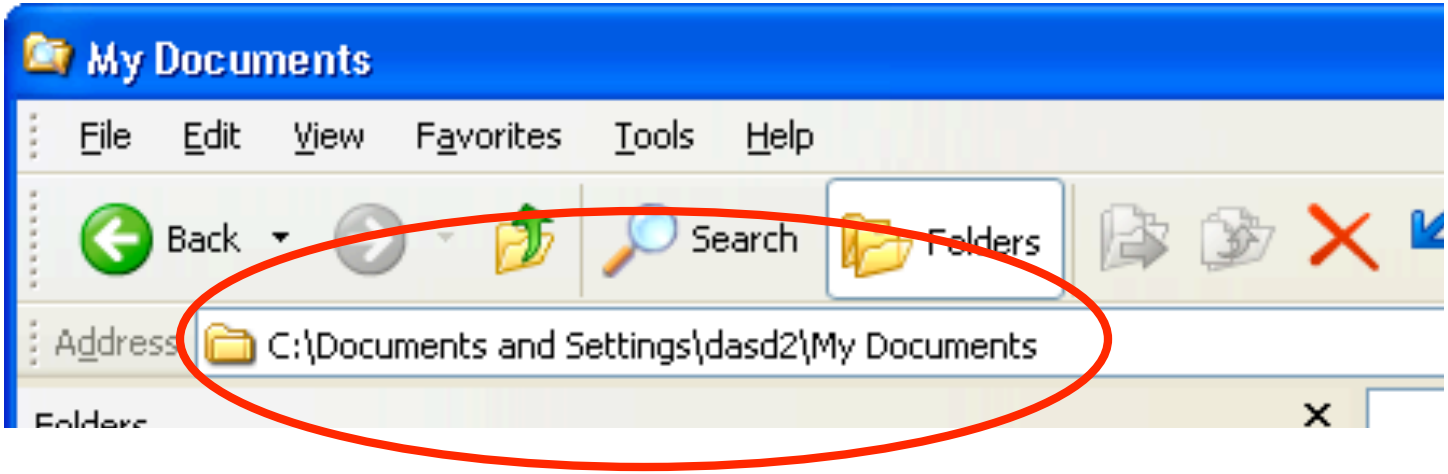


Windows Explorer



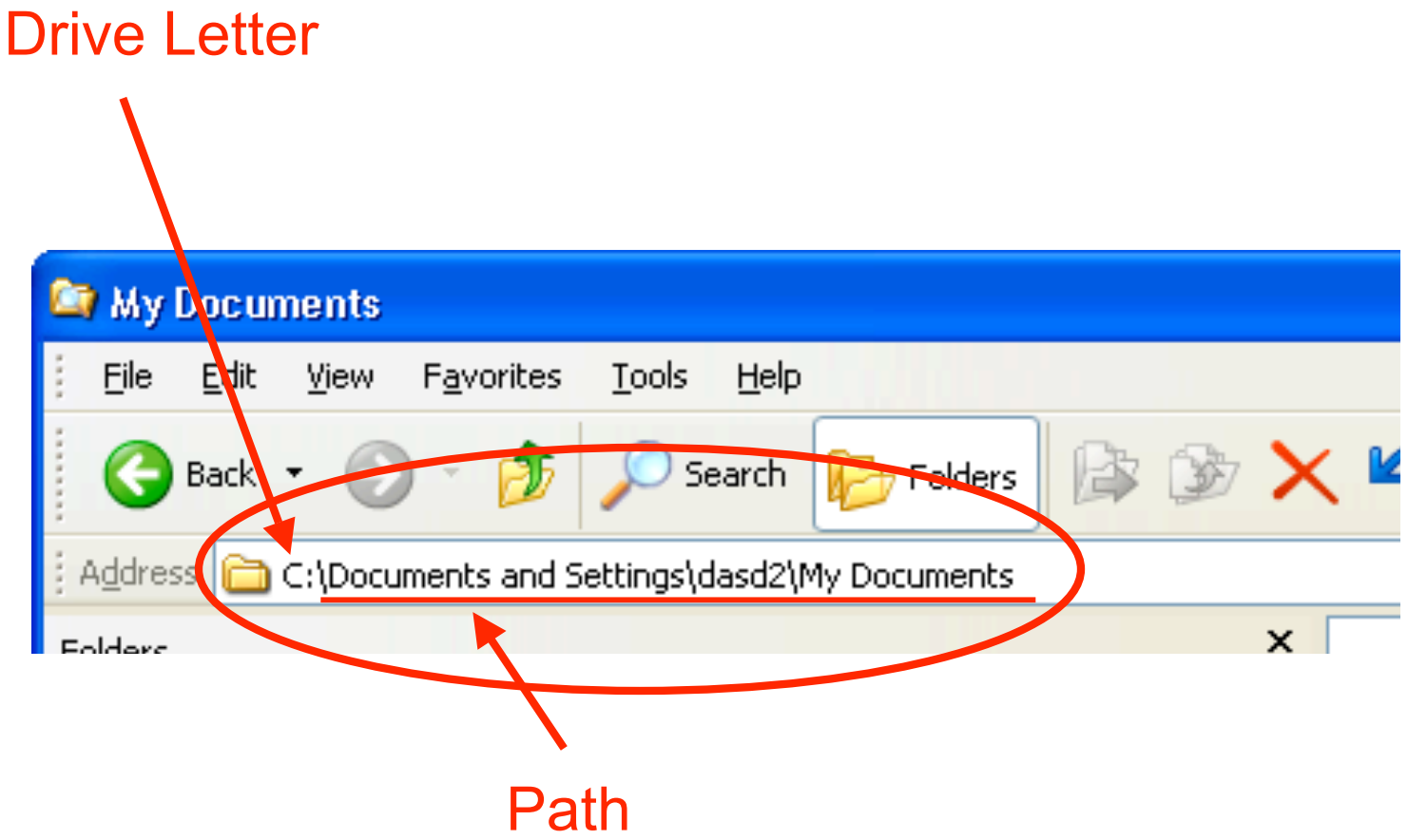
Windows Explorer

- We can click around and explore all the files / folders our computer has access to.
- The exact location of the highlighted folder is given in the address bar.



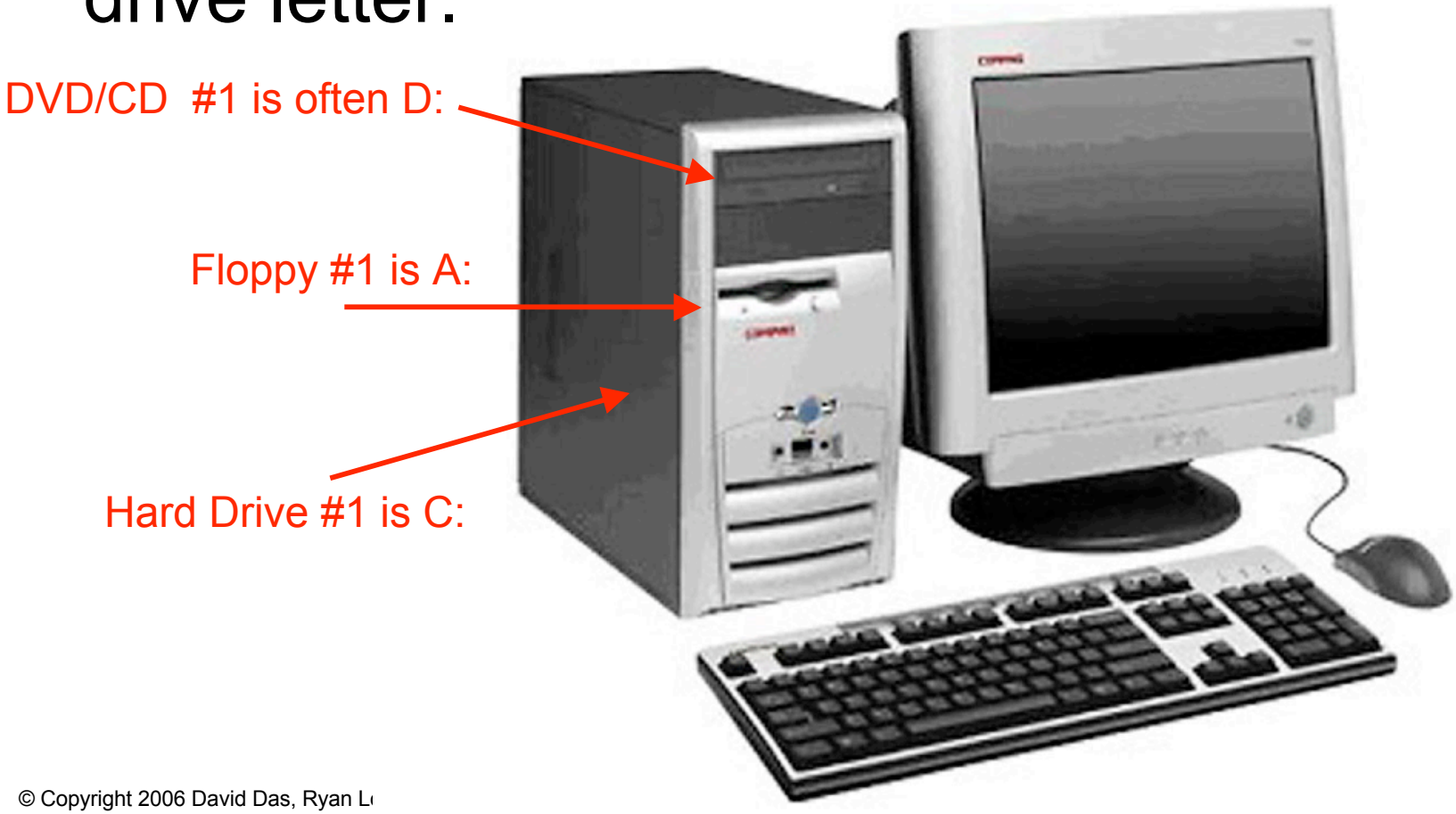
Windows Explorer

- Lets examine the address:



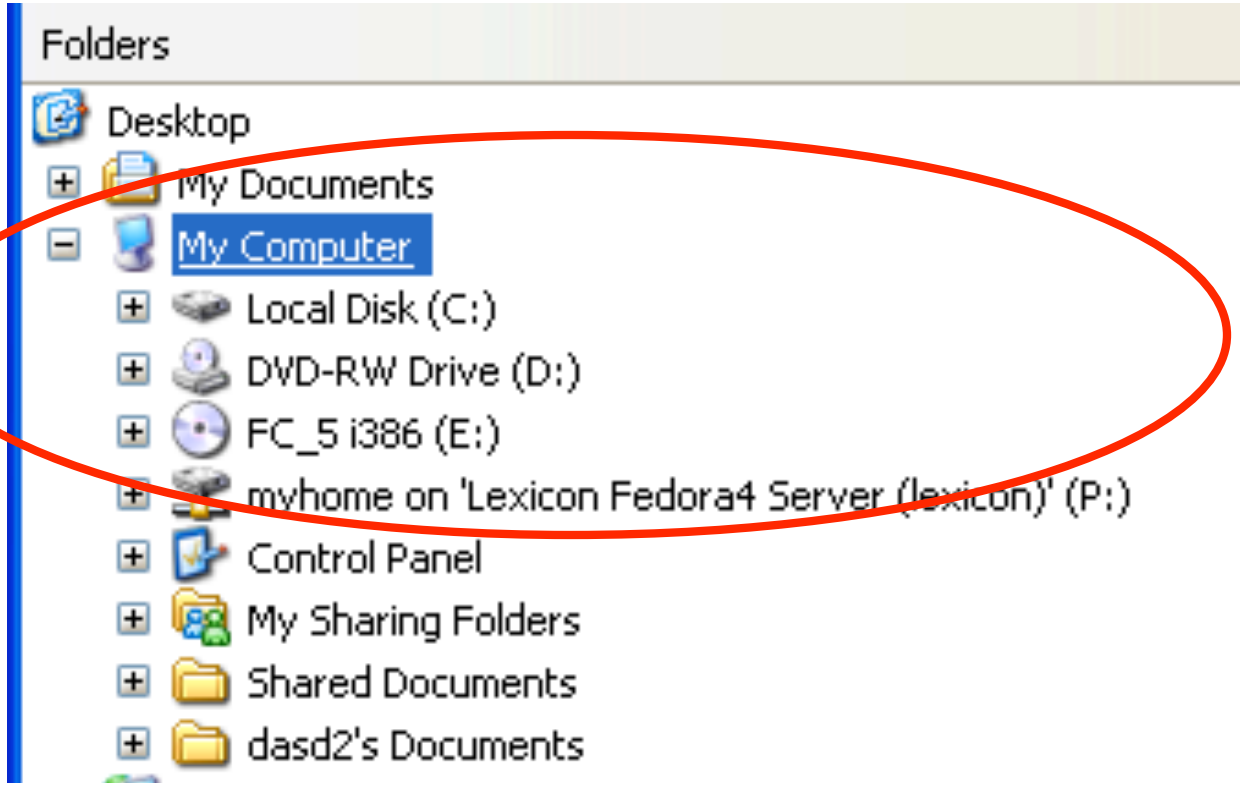
Drive Structure

- Each hardware device has at least one drive letter.



File Structure

- These drives are visualized in Windows Explorer on the left

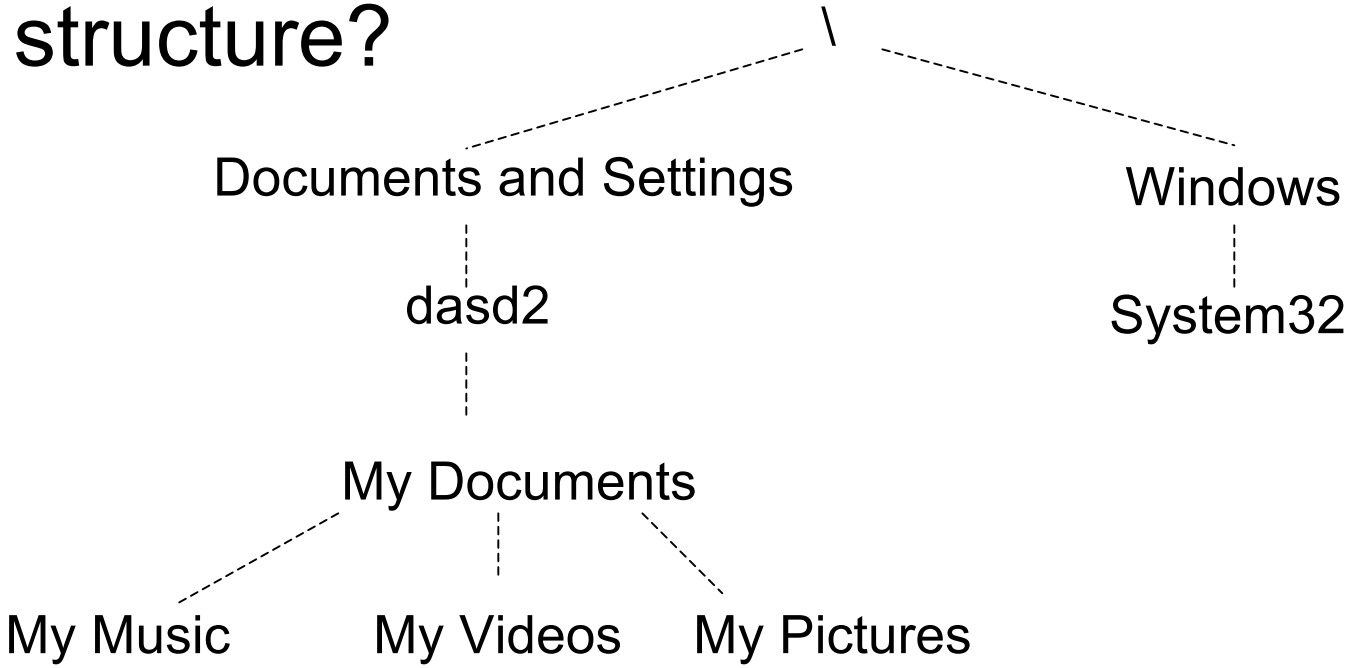


Drive Structure

- Each device has a main directory called a root directory/folder and is represented by a \.
- Sub-folders are branches off the root directory separated by \.

Windows Explorer

- Notice how the folders are in a tree structure?

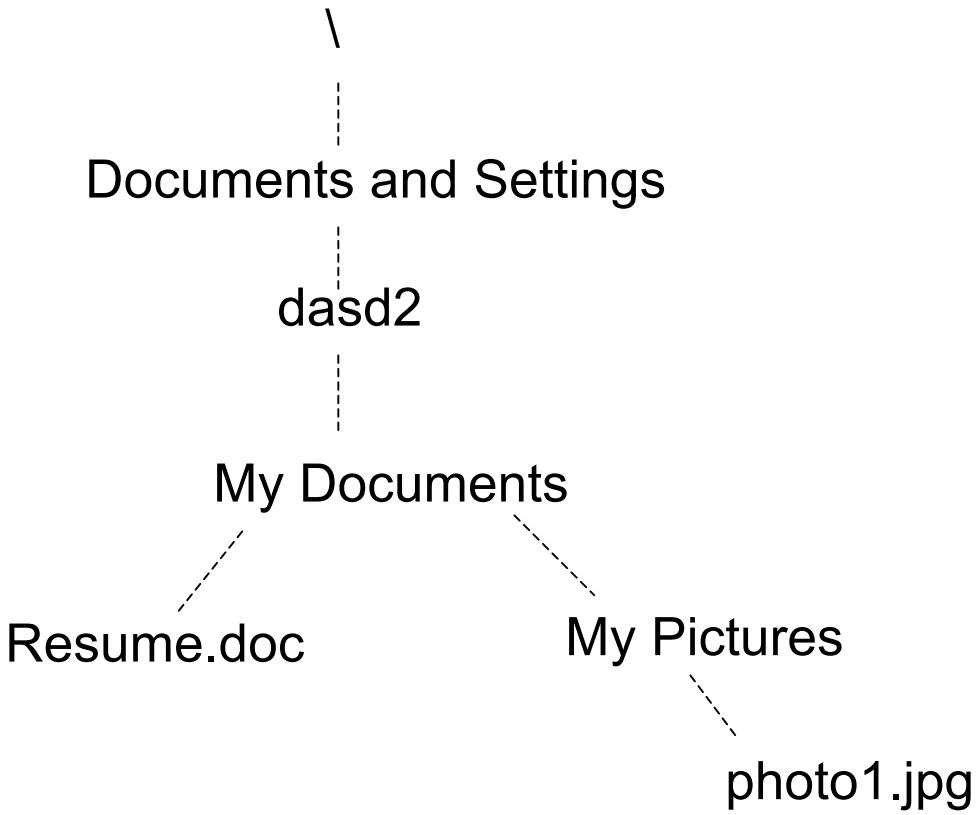


Drive Structure

- All files on your computer can be referenced by a full filename.
- This name includes:
  - The Drive Letter
  - The Path
  - The Filename
- This name is case sensitive.

```
C:\Documents and Settings\dasd2\My Documents\Resume.doc
C:\Documents and Settings\dasd2\My Documents\
    My Pictures\photo1.jpg
```

Drive Structure




```
C:\Documents and Settings\dasd2\My Documents\Resume.doc
C:\Documents and Settings\dasd2\My Documents\
      My Pictures\photo1.jpg
```

File I/O

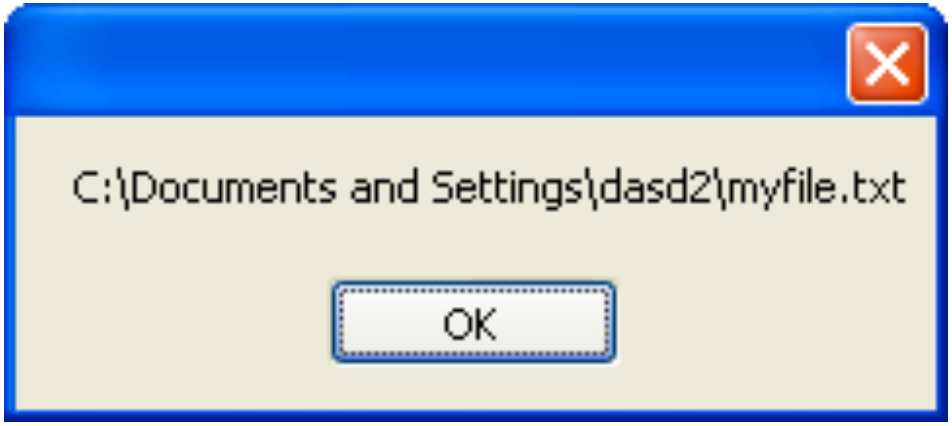
why do we want to?

- Putting a fully qualified name into a string poses one problem. The '\' character cannot be put directly into a string.
- In C#, the '\' character is known as an escape character. It is used to put special, non-keyboard characters into a string.
- As an example: in order to put a tab into a string, the code "\t" must be entered.

File I/O

- In order to put an actual '\' into a string instead of using it as an escape character, two \\'s must be entered.

```
MessageBox.Show("C:\\Documents and Settings\\dasd2\\myfile.txt");
```

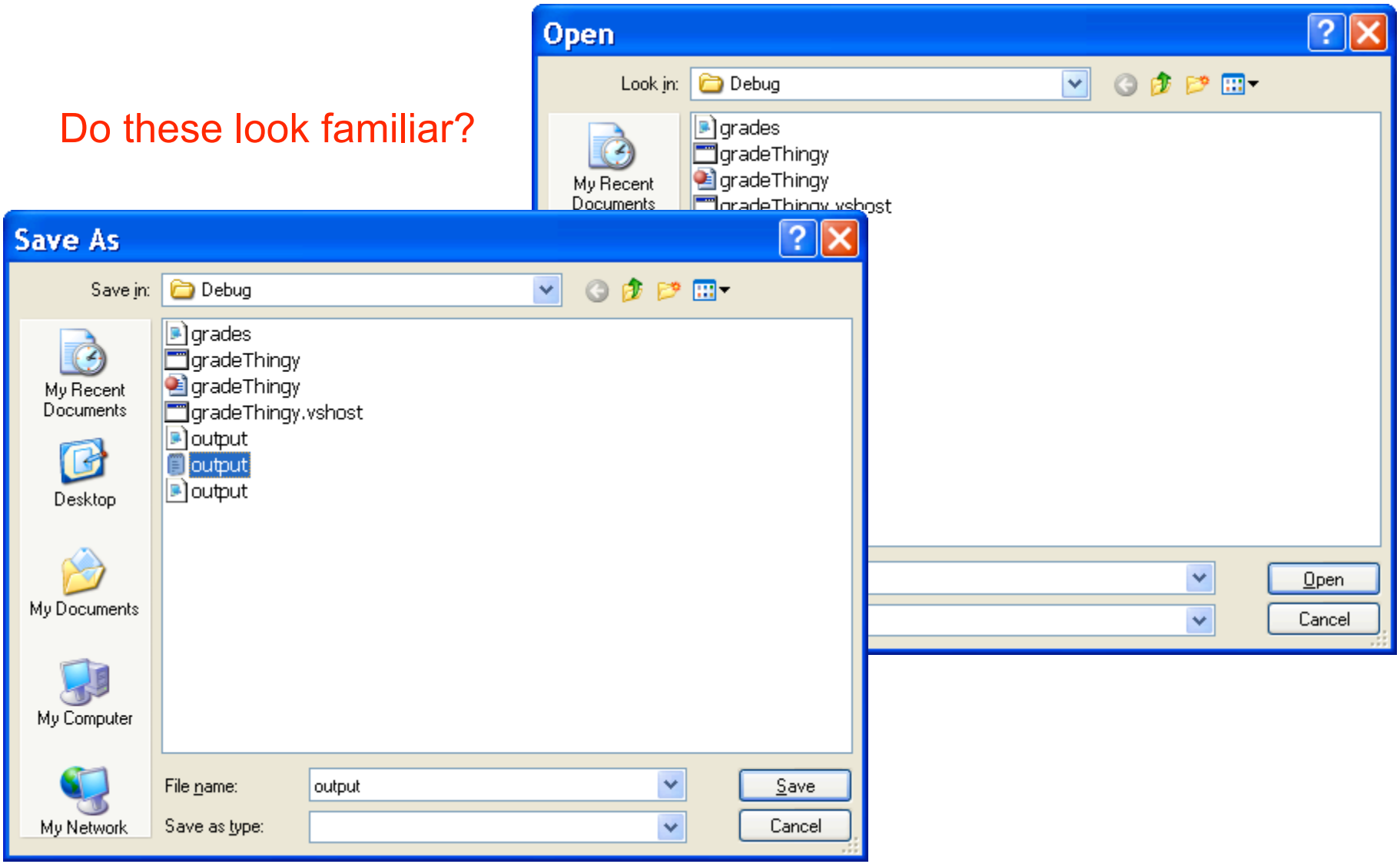


File I/O

- When programming, it is also possible to use a GUI tool to select a file and create a string from its fully qualified filename.

File I/O

Do these look familiar?



File I/O

- A file dialog box will produce the string of a fully qualified filename for a selected file.
- The OpenFileDialog allows a user to select a file that exists.
- The SaveFileDialog allows a user to specify a filename. If the file already exists, the dialog will ensure the user knows that this file will be overwritten.

File I/O

- This code creates two new dialog boxes.
  - of ⇒ An OpenFileDialog
  - sf ⇒ A SaveFileDialog

```
OpenFileDialog of = new OpenFileDialog();
SaveFileDialog sf = new SaveFileDialog();

of.ShowDialog();
sf.ShowDialog();
```

File I/O

- The dialog boxes are now shown. The OpenFileDialog will appear first.
- The user can make a file selection.

```
OpenFileDialog of = new OpenFileDialog();
SaveFileDialog sf = new SaveFileDialog();

of.ShowDialog();
sf.ShowDialog();
```

File I/O

- The filenames will now be available in:
  - of.FileName;
  - sf.FileName;

- Right now, it is important that you know how to use this code. It is not important that you know why this code works. This will all become clear soon.

1D04 Grade Distribution - Revised

```
void findGradeDistribution()
{
   int[] gradeDistribution = new int[13];
   string inputString;
   int inputInt;
   StreamReader inStream;
   StreamWriter outStream;
   OpenFileDialog of = new OpenFileDialog();
   SaveFileDialog sf = new SaveFileDialog();

   of.ShowDialog();
   sf.ShowDialog();

   inStream = new StreamReader(of.FileName);
   outStream = new StreamWriter(sf.FileName);
. . .
```

1D04 Grade Distribution - Revised

```
. . .
   inputString = inStream.ReadLine();
   while (inputString != null)
   {
      inputInt = Convert.ToInt32(inputString);
      gradeDistribution[inputInt]++;
      inputString = inStream.ReadLine();
   }

   for (int i = 0; i < gradeDistribution.Length; i++)
   {
      outStream.WriteLine("Grade Point " + i + " # "
                          + gradeDistribution[i]);
   }
   outStream.Close();
   inStream.Close();
}
```

Parsing File Contents

- Suppose we have a name and a grade on a single line in a file separated by a comma.
- What would it take to turn this file into an array of names and an array of grades?

```
Tyler,12
Austin,4
Makayla,6
Jasmine,11
```

Parsing Algorithm

- Use a string array called name for names and an integer array called grade for grades.
- Start at the beginning of the file and the beginning of the arrays. [0]
- Read one line of the file and separate into name and grade (still magic).
- Store name and grade into their arrays.
- Move onto the next line of file.
- Repeat until file is completely read.

Base Code

```
void difficultFile()
{
    StreamReader inStream = new StreamReader("difficult.xxx");
    string inputString;
    string [] name = new string[500];
    int [] grade = new int [500];

    inputString = inStream.ReadLine();
    while (inputString != null)
    {
       // Parse the file into something useful
       // still magic
       inputString = inStream.ReadLine();
    }
    inStream.Close();
}
```

Parsing Strings

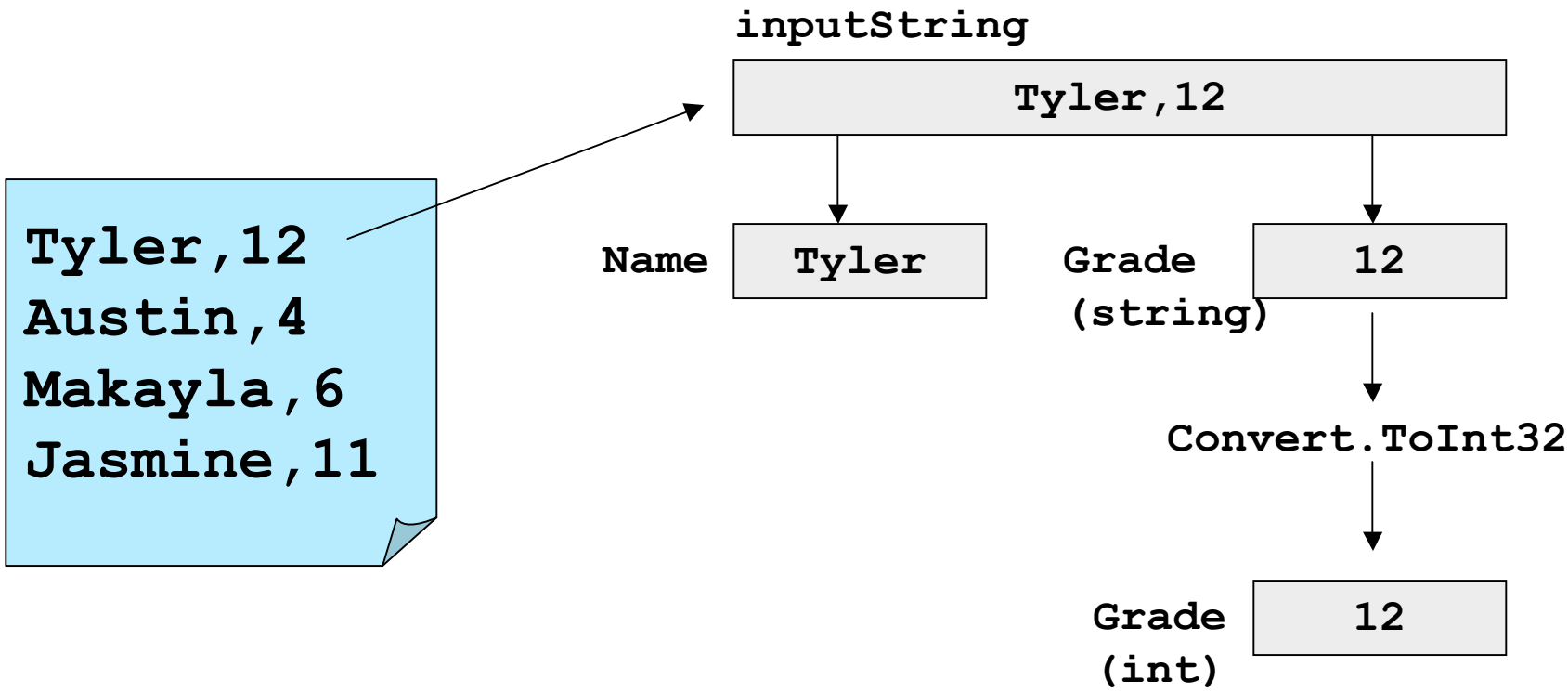
- We know how to read a line from a file using ReadLine. It comes in as a string.
- So, we need to learn some advanced string manipulation techniques so that we can dice up each line appropriately.

```
Tyler,12
Austin,4
Makayla,6
Jasmine,11
```

String Dicing Algorithm

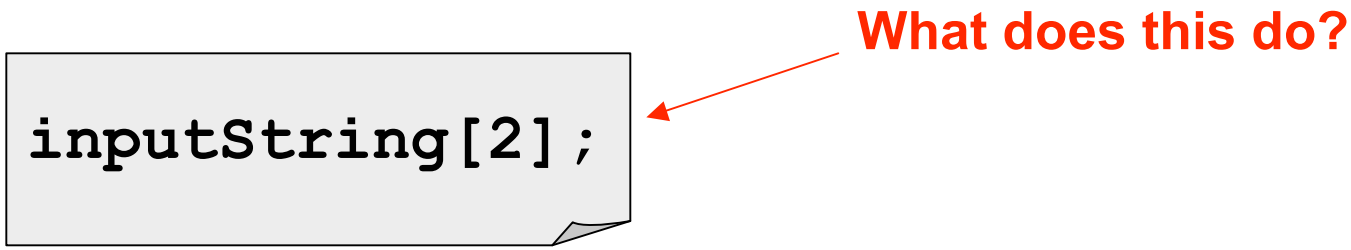
- Find the location of the comma in inputString.
- Split the string based on where the comma is so that the name is to the left of the comma and the grade is to the right.

Parsing Strings



Parsing Strings

- Strings elements are similar to arrays.
- Each character in a string can be accessed by an index.
- Unlike normal arrays, we cannot modify the contents of the string at an index.



Parsing Strings

- We can loop through a string looking at each element until we find a comma.

```
int i = 0;
while((i < inputString.Length) &&
      (inputString[i] != ',' ))
{
   i++;
}
```

What is the value of i when this loop finishes executing?

Parsing Strings

- There is an easier way.
- A string has an IndexOf method.

Parsing Strings

```
string inputString = "Tyler,12";
int index;

index = inputString.IndexOf(',');
```

```
index = 5
```

Parsing Strings

- IndexOf() can take either a string or a character and return the index of it in a string.
  - inputString.IndexOf("blah");
  - inputString.IndexOf(',');
- If this element is not found in the string, IndexOf() returns -1.

Parsing Strings

- Now, to split up the string.
- We've identified the index of the point at which we want to split the line.
- What method may be useful in this situation?

Parsing Strings

- Substring:
  - The first parameter is the index to start copying the source string from.
  - The second parameter is the length of the substring to be copied.

```
name = inputString.Substring(0, index);
grade = Convert.ToInt32(inputString.Substring(
  index + 1, inputString.Length - index - 1));
```

Parsing Strings

- The conversion for the last half of the string is painful.
- It should be done without calculating where the end of the string is.

```
name = inputString.Substring(0, index);
grade = Convert.ToInt32(inputString.Substring(
  index + 1, inputString.Length - index - 1));
```

Parsing Strings

- Substring with 1 parameter:
  - The parameter is the index to start copying the source string from. This will copy all the characters in the string from this point until the end.

```
name = inputString.Substring(0, index);
grade = Convert.ToInt32(inputString.Substring(
   index + 1));
```

Parsing Strings

```
string inputString = "Tyler,12";
int index;
int grade;
string name;

index = inputString.IndexOf(',');
name = inputString.Substring(0,index);
grade = Convert.ToInt32(
          inputString.Substring(index + 1));
```

Full Example

```
void difficultFile(){
    StreamReader inStream = new StreamReader("difficult.xxx");
    string inputString;
    string [] name = new string[500];
    int [] grade = new int [500];
    int index;
    int i = 0;

    inputString = inStream.ReadLine();
    while (inputString != null){
       index = inputString.IndexOf(',');
       name[i] = inputString.Substring(0, index);
       grade[i] = Convert.ToInt32(
                     inputString.Substring(index+1));
       i++;
       inputString = inStream.ReadLine();
    }
    inStream.Close();
}
```

Parsing Strings

- What about the following situation? How would we determine where the second comma is?

```
string inputString = "Das,Dave,6";
```

Parsing Strings

```
string inputString = "Das,Dave,6";
int comma1, comma2;

comma1 = inputString.IndexOf(',');
comma2 = inputString.IndexOf(',', comma1 + 1);
```

Parsing Strings

- IndexOf can take a second parameter.
- This is the index that the indexOf method will begin at.
- If we start at the index directly after the first comma, we'll find the index of the second comma.

```
comma2 = inputString.IndexOf(',', comma1 + 1);
```

Parsing Strings

- It's also possible that there is garbage information in a file.
- There may be extra white space before or after a name.
- You have all the tools you need to deal with it.
- How would you?

```
Tyler,12
Austin  ,4
Makayla ,6
 Jasmine,11
```