COMP3015 Data Communication and Networking
File Management

Windows Commands

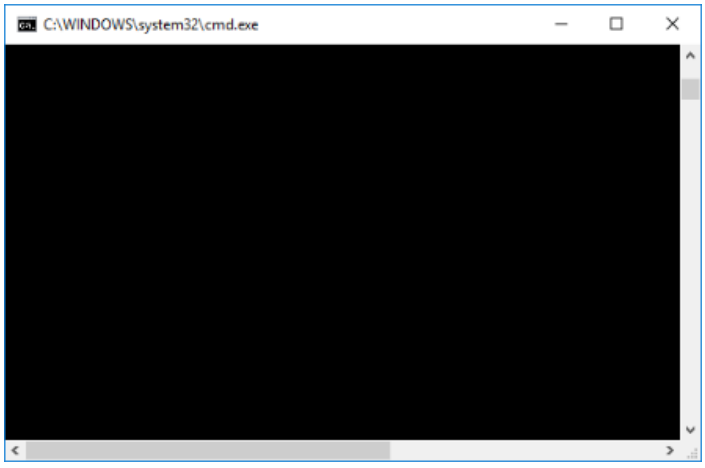
The command line is a program that provides a text interface for accessing the operating system. From the command line, you can check the information about your system and manage the files and folders on your computer. With the command line, you can perform almost all the same tasks that can be done with GUI. However, the command line can be performed quicker and can be easier to automate (batch).

Many commands can be used in the command line. Each operating system has its own set of commands. In this lab, we discuss some commonly used and useful commands for managing files and directories on macOS, Linux, and Microsoft Windows.
Windows has the command line tool too, but the commands are not the same as Linux's commands.

The name of the command line tool of Windows is "Command Prompt". To open the command prompt, we need to:

1. Hold the "Windows" key and then press "R".

2. Type "CMD" in the text field and click "OK".



Windows Commands

DIR

"DIR" means directory that is used for listing the files and subdirectories in a directory (folder).
Let us type the following in the command prompt:

`dir`⏎	List all files/subdirectories in the current directory.
`dir .`⏎	Same as above.
`dir ..`⏎	List all files/subdirectories in the parent directory.
`dir \`⏎	List all files/subdirectories in the root directory.
`dir\windows`⏎	List all files/subdirectories in the specific path. In this example, the path is "\windows".

⏎ means press ENTER

The following is the sample output of "dir \windows":

```
C:\Users\helloworld>dir \windows
 Volume in drive C has no label.
 Volume Serial Number is 8012-44D1

Directory of C:\windows

08/23/2020  12:48 PM    <DIR>          .
08/23/2020  12:48 PM    <DIR>          ..
04/12/2018  07:38 AM    <DIR>          addins
07/23/2018  10:20 AM    <DIR>          appcompat
04/12/2018  07:34 AM            67,072 bfsvc.exe
08/29/2019  05:12 PM            31,423 comsetup.log
07/18/2018  01:57 PM             2,269 DDACLSys.log
08/29/2019  04:10 PM            24,768 diagerr.xml
08/29/2019  04:10 PM            24,768 diagwrn.xml
09/16/2016  04:37 PM            10,695 DirectX.log
…
```

Directories

Files

Last modified | Directory indicator | File size | name

DIR: Wildcard

You can use wildcard characters (* or ?), to represent one or more characters of a file name and to display a subset of files or subdirectories.

* means zero or more characters.

? means zero or one character.

For example:

`dir *.txt⏎`	List files/subdirectories in the current directory with the name end with ".text". (Extension name is "text") Such as abc.txt, zyz123.txt, etc.
`dir *.txt?⏎`	List files/subdirectories with the name end with ".txt" and followed by zero or one character. Such as abc.txt, abc.txt1, xyz123.txt2, etc.
`dir A?B*⏎`	Such as A0B, A1B23.txt, etc.

DIR: Name with White Spaces

If you want to use the DIR command with the name includes white spaces, you need to use double quotation marks to quote the name. For example, we want to list all files/subdirectories in the C:\Program Files directory. We need to use:

```
dir "C:\Program Files"⏎
```

Note that Microsoft Windows is a case-insensitive system. The following command will return the same output as above.

```
DIR "c:\program files"⏎
```

DIR: Switch

We can change the output format by providing different switches. Let us try the followings:

`dir /AD⏎`	List all subdirectories in the current directory.
`dir /AH⏎`	List all hidden files in the current directory.
`dir /W⏎`	Use a wide list to output the information.
`dir /OD⏎`	Sort the output by date in ascending order.

`dir /ON⏎`	Sort the output by name in alphabetic order.
`dir /P⏎`	Pause after each screenful of information. Press any key to see the next screen.
`dir /?⏎`	Display the details of the DIR command.

We can combine multiple switches in a single DIR command. For example, we need to "wide list" subdirectories in \windows and sort the output by name. So, we can use:

```
dir /W /AD /ON \windows⏎
```

Changing Current Drive – C:, D:, …, Z:

Windows has the "drive" concept. You will find a new drive when you can plug a USB thumb drive to the USB port of your computer. You will find multiple drives too if your computer has multiple hard disks installed. If we want to manage the files/directories on different drives, you need to change the current drive to the specific drive.

The default drive normally is C drive because your home directory is on C drive. And, by default, the current directory is your home directory when you open a command prompt. "Home directory" is a directory created for your user account.
If you want to change the current drive to Z drive (assume that your computer has Z drive), you only need to type:

```
Z:⏎
```

CHDIR (CD)

"CHDIR" or "CD" means change directory. It is used for changing the current directory to another. The following is the usage of the CD command:

`cd⏎`	Display the path of the current directory.
`cd Document⏎`	Change the current directory to

`cd /D c:\windows⏎`	Same as
	`C:⏎`
	`cd \windows⏎`

MKDIR (MD)

"MKDIR" or "MD" means make directory. It is used for creating a new directory. I suggest you change the current directory to the Download directory under your home directory before you try the "MD" command. You may use the following command to do so:

```
cd /D %HOMEDRIVE%\%HOMEPATH%\Downloads⏎
```

%HOMEDRIVE% is the system environment variable that stores the drive of your home directory. **%HOMEPATH%** stores the path of your home directory.

After changing the current directory, you may type `cd⏎` to verify it. Now, we can try the following commands:

`md hello⏎`	Create a subdirectory named hello under the current directory if it does exist.

`md hello\world⏎`	Same as:
	`md hello⏎`
	`cd hello⏎`
	`md world⏎`
	Then, go back to the original directory

`md Z:\hello⏎`	Create a subdirectory named hello under the root directory of Z drive.

Note that the "MD" command will not change the current directory.

ECHO

"ECHO" is used for outputting a text message. Let us try to the following:

```
echo hello world⏎
```

You will see "hello world" outputted immediately in the command prompt.
You can change the output to a file instead of the command prompt by using the > sign. Try the following:

```
echo hello world > abc.txt⏎
```

The text is printed in the file named helloworld.txt. You can find the file in Windows Explorer and check it.

You can use two > signs to append text to the file. For example:

```
echo nice to meet you >> abc.txt⏎
```

TYPE

"TYPE" is the command to read the content from a text file. To use it, you only need to put the name of the text file following "TYPE".

```
type abc.txt⏎
```

COPY

"COPY" is the command for copying files to another location. Let us try the following commands:

`copy abc.txt xyz.txt⏎`	Assume that the source file named *abc.txt* is stored in the current directory. Copy the source file to the destination file named *xyz.txt*. After copying the contents of both source and destination files are the same. If the destination file exists, the "COPY" command will ask for the action – overwrite or not.
`copy *.txt hello⏎`	Assume that a subdirectory named *hello* exists in the current directory. Copy all files with extension name "txt" to the destination directory named hello, in the current directory.
`copy *.txt hello⏎`	**If there is no subdirectory named hello in the current directory, the "COPY" command will make a file named *hello* and the content of the file will be the last file handled by the "COPY" command.
`copy /Y *.txt hello⏎`	Assume that a subdirectory named *hello* exists in the current directory. Copy all files with extension name "txt" to the destination directory named hello, in the current directory. If the destination files exist, the "COPY" command will not ask for the action and the existing files will be overwritten.

MOVE

"MOVE" is used for moving files to another location. The syntax is the same as the "COPY" command, but the source files will be deleted.

```
move *.txt hello\world\⏎
```

The "MOVE" command can be used for renaming a file or directory. For example, we move all txt files from the hello directory to *hello/world* directory.

```
move hello.txt world.txt⏎
```

DEL

"DEL" is used for deleting files. Note that the "DEL" command will not move your deleted files/directories to *Recycle Bin*. Therefore, you need to consider before acting.
To delete a file, you only need to put a file following "DEL". For example, we delete *abc.txt* in the current directory.

```
del abc.txt⏎
```

You may use a wildcard for deleting multiple files or you can put multiple file names following "DEL". For example, we delete *abc.txt* and *xyz.txt* in the current directory.

```
del abc.txt xyz.txt⏎
```

If a file is marked as read-only, the "DEL" command will not act. You may add /F switch to force delete the file.

```
del /F abc.txt⏎
```

You can use the following command to delete all files and subdirectories in the specified directory. For example, we delete everything stored in the *hello* subdirectory under the current directory.

```
del /S /F hello/⏎
```

File class in Java

In Java, the File class provides file and directory manipulation. It has many methods for retrieving information of files or directories, just like the commands provided by the Command Prompt.

The following code should the detail information of a file or directory:

```
public static void getInfo(String filename) throws IOException {
   File file = new File(filename);
   System.out.println("name : " + file.getName());
   System.out.println("size (bytes) : " + file.length());
   System.out.println("absolute path? : " + file.isAbsolute());
   System.out.println("exists? : " + file.exists());
   System.out.println("hidden? : " + file.isHidden());
   System.out.println("dir? : " + file.isDirectory());
   System.out.println("file? : " + file.isFile());
   System.out.println("modified (timestamp) : " + file.lastModified());
   System.out.println("readable? : " + file.canRead());
   System.out.println("writable? : " + file.canWrite());
   System.out.println("executable? : " + file.canExecute());
   System.out.println("parent : " + file.getParent());
   System.out.println("absolute file : " + file.getAbsoluteFile());
   System.out.println("absolute path : " + file.getAbsolutePath());
   System.out.println("canonical file : " + file.getCanonicalFile());
   System.out.println("canonical path : " + file.getCanonicalPath());
}
```

The following are the sample results:

```
> D:/
                  name :
           size (bytes) : 8192
         absolute path? : true
                exists? : true
                hidden? : true
                   dir? : true
                  file? : false
   modified (timestamp) : 1535685446303
              readable? : true
              writable? : true
            executable? : true
                parent : null
         absolute file : D:\
         absolute path : D:\
        canonical file : D:\
        canonical path : D:\
```

```
> D:\img1.jpg
                  name : img1.jpg
           size (bytes) : 844818
         absolute path? : true
                exists? : true
                hidden? : false
                   dir? : false
                  file? : true
   modified (timestamp) : 1471969391181
              readable? : true
              writable? : true
            executable? : true
                parent : D:\
         absolute file : D:\img1.jpg
         absolute path : D:\img1.jpg
        canonical file : D:\img1.jpg
        canonical path : D:\img1.jpg
```

Getting File and Subdirectory List

If the **File** object associates with an existing directory, we can use its **list()** or **listFiles()** method to get a list of the files in the directory.

- **list()** returns a String array that contains the names of the files and subdirectories of the specified directory.
- **listFiles()** returns a File array that contains the file objects of the files and subdirectories.

The following code puts names of the files and subdirectories of the specified directory in a single string.

```
public String listDir(String pathname) {

   File path = new File(pathname);

   String[] filelist = path.list();

   return String.join("\n", filelist);

}
```

The following example code shows the sizes of the individual files in the specific directory:

```
public String listDir2(String pathname) {
   File path = new File(pathname);
   File[]  files = path.listFiles();
   ArrayList<String> info = new ArrayList<>();

   for(File f : files) {
      if (!f.isDirectory()) {
         info.add(
            String.format("the size of %s is %d bytes",
            f.getName(), f.length() ));
      }
   }

   return String.join("\n", info);
}
```

Creating Directory

The **mkdir()** and **mkdirs()** methods are used to create subdirectories.

- **mkdir()** – single level
- **mkdirs()** – multiple levels

Imagine that we are going to create directors so that we will have the path *c:\abc\xyz\123*. Assume that the directories *abc*, *xyz*, and *123* do not exist currently. We use the following statements to create the directories:

```
new File("/abc").mkdir();
new File("/abc/xyz").mkdir();
new File("/abc/xyz/123").mkdir();
```

Or we can use a single statement as follow:

```
new File("/abc/xyz/123").mkdirs();
```

Note that **mkdir()** and **mkdirs()** will not throw errors if the specific directory exists.

Deleting File/Directory

The **delete()** method is used to delete file or directory. It returns true if the deletion succeeded. It will not throw exceptions even if the file path does not exist.

The **delete()** method may not act and return false if the target file/directory is locked. Therefore:

- To delete a file, you need to ensure that the file is not currently opened by any applications.

- To delete a directory, you need to ensure the followings.
  1. The target directory is an empty directory.
  2. The other applications (including Window Explorer) are currently accessing the target directory.

```
new File("/abc").delele();
```