Available selections in class\_week2

1. **Unix Filesystem Hierarchy**

* The Linux OS has a 'hierarchical' file system which means that directories can contain other directories or files.
* The resulting structure may be drawn as a 'tree diagram', which is usually flipped so that the root directory is at the top:  
    
                  / (root)  
                  |  
          --------------------------------  
          |          |         |         |  
         bin        dev       usr       etc  
                                         |  
                                  ---------------  
                                  |      |      |  
                                  A      B      C
* In the Linux and Unix OS, the storage device such as the hard disk contains directories. Directories are actually files that are used to store other files. This allows the storage device to be organized and prevent confusion.
* The root directory (denoted by a slash '/') is the beginning (or master) directory.
* All other directories are related to the root directory -- they are contained, directly or indirectly, within the root directory.
* Directory Terminology
* Directory : A 'directory file' used to store other files
* Sub-directory : A directory file DIRECTLY dependent on previous or 'parent' directory. Also called a 'child directory'.
* Parent Directory : A directory file that contains a sub-directory.
* Normally a subdirectory is referenced relative to its parent directory.  For example, in the previous tree diagram, 'A' is a subdirectory of 'etc'.
* Important things to note about Unix file hierarchies
* There are no drive letters. All of the disk drives are 'mounted' into single hierarchy.
* In Unix, file names are used for many types of resources that you would not usually think of as files, such as directories, devices, communication between programs, status information, and network connections (remember that 'everything is a file').
* Naming Files (and Directories) : The maximum length of a file name, whether ordinary files or directories, varies from system to system. The sizes can range from 14 to 255 characters.  Most characters can be used in filenames, but for simplicity,

**you should stick to:**

* Uppercase or lowercase letters
* Numbers
* Underscore '\_'
* Period '.'
* Comma ','

\*\*A period BEFORE a filename 'hides' the file.   All file names are case sensitive.

1. **Pathnames :** A pathname is used to specify the location of a file or directory.  Pathnames are used when issuing Linux commands when working with directories and files.  There are **three categories** of pathnames

* Pathname is a list of names that will lead to a file
* Essentially they are directories, but a file name itself is a path as well
* The concept of a pathname relates to every operation system including Unix, Linux, MS-DOS, MS-Windows. Apple Macintish
* Examples
* Directory pathname : /home/username/ics124/assignments/
* File pathname : /home/username/ops224/assignments/assn1.txt
* **Absolute Pathname** : location reference starting from root '/
* An absolute pathname specifies how to **find a file**, **starting at the root directory** and working your way down to a specific file. Absolute pathnames **always start with a slash '/'** character.
* An absolute pathname consists of a list of directories separated by slashes '/' and ending with the particular file or directory to which you are referring.
* For example, the absolute pathname:  
    
  /etc/system/config/ejb.conf  
    
  Specifies the file 'ejb.conf' stored withing the 'config' directory which is within the 'system' directory within the 'etc' directory, which is in the root directory.
* **Relative Pathname** : location reference starting from the 'current directory'  doesn’t start with /
* Unix systems employ the concept of a current directory, which you can set to any directory in the hierarchy.
* A relative pathname does not start with a slash '/' or a tilde '~' and specifies how to find a particular file or directory starting from the current directory.
* If the current directory is set to '/etc/system', then these pathnames are equivalent:  
    
  Relative Pathname     Absolute Pathname  
    
  foo.txt         /etc/system/foo.txt  
  conf/current    /etc/system/conf/current  
  old/red/blue    /etc/system/old/red/blue
* When using **relative pathnames**, these symbols are available:  
    
  **.**   This (current) directory  
    **..** Parent directory ( up one directory level)
* If the current directory is set to '/etc/system', then these pathnames are equivalent:  
    
  Relative Pathname                        
    
    .                            /etc/system  
    foo.txt                      /etc/system/foo.txt  
    ./foo.txt                    /etc/system/foo.txt  
    ../foo.txt                   /etc/foo.txt  
    ../../home/joe/text     /home/joe/text

[e.g] Change to another directory branch from parent directory

cd ../ipc144

Copy sample.c file from joe.professor’s home directory to your current directory

cp ../joe.professor/uli101/sample.c

* **Relative-to-Home Pathname**: location reference starting from your 'home directory' Start with ~
* Each user of a Unix system is assigned a home directory by the system administrator. On many Unix systems, this is /home/userid (where 'userid' is the user's account ID), but home directories may be located anywhere in the hierarchy.
* The home directory is provided for you to store your personal files and directories. On this computer, you have been assigned the home directory       
  '/home/ayesha.manzer'
* Important:

1. Note that /home is NOT the home directory!
2. Note that your home directory is NOT necessarily the same as your current directory. Your current directory changes as you move around the file hierarchy, but your home directory stays the same.

* Relative-to-Home pathnames start with a tilde '~'.

1. The tilde ~ is replaced by your home directory (typically /home/your.account/) to make the pathname absolute.
2. If you specify a tilde followed by a slash, the pathname is relative to your home directory.
3. You can also specify a pathname relative to another user's home directory by starting your pathname with '~userid/'.

[e.g]Your home directory is /home/ayesha.manzer, so these pathnames are equivalent:  
  
Relative-to-Home        Absolute Pathname  
  ~/lastyear.txt      /home/ayesha.manzer/lastyear.txt  
  ~/unx122/notes      /home/ayesha.manzer/unx122/notes

If your friend's home directory was '**/usr/friend**' and their user ID was '**frnd**', these pathnames would be equivalent:  
  
Relative-to-Home        Absolute Pathname  
  ~frnd/test           /usr/friend/test  
  ~frnd/2002/diagram   /usr/friend/2002/diagram

|  |
| --- |
| **Could you explain me the difference between absolute path and relative path?**  /export/home/heden/rhost  The above is said to be absolute path.Is it because the forward slash?  In Windows, we talked about a single path. There are no two paths in Windows.  What are those absolute and relative paths?  **Quote:**  Pretty much - by putting that / at the start, you're saying "Start at the root directory, and work along"  If you don't put the / at the front, you're saying "Start from the current directory"  If you put ../ at the front, you're saying "Go back one directory and then start looking"  And if you put ~/ at the front, you're saying "Look in my home directory"  So, to sum up, let's say you're in the directory /etc/wibble  If you say cd /foo/bar you're asking to go to the directory /foo/bar  If you say cd foo/bar you're asking to go to the directory /etc/wibble/foo/bar  If you say cd ../foo/bar you're asking to go to the directory /etc/foo/bar  And if you say cd ~/foo/bar you're asking to go to the directory /home/yourname/wibble/foo/bar  export /home/heden/rhost is absolute  export home/heden/rhost is relative to the current working directory  **./ refers to the current directory**  / **= root directory**  ./ **= current directory**  ../ **= parent directory**  You often don't have to bother with ./ - if you're in /foo and you want to switch to /foo/bar, you can use either:  cd /foo/bar  cd ./bar  or just  cd bar so naturally this is the one you'd usually use  When the ./ does become useful is for things like moving files to the current directory,  mv /foo/bar/\* ./ |

1. **Ambiguous Pathnames & Globbing**

* Unix provides the 'ambiguous pathnames' feature to permit you to specify a number of files at once.  There are three 'wildcard' symbols used in ambiguous pathnames:

File name expansion also called ambiguous file references, metacharacters, wild card characters, and filename generation characters  
  
**\***         : matches zero or more of any character (star/asterisk)  
**?**          : matches exactly one of any character (question mark)  
**[class]**: matches exactly one character from the class (character class)

* **The Astrisk** : The astrisk matches zero or more characters, like this:  
    
  Pattern         Matches         Doesn't Match  
    
   \*             (anything!)  
    
   a\*b            ab               A12  
                  a2314234b        able  
                  all\_the\_tab      resub  
    
   \*txt         file.txt         txtfile ( start the txt does not matching)  
                  file\_txt         bastxtyot( end of the txt doesn’t matching)  
                  texttxt
* **The Question-Mark** : The question-mark matches exactly one of any character, like this:  
    
  Pattern           Matches         Doesn't Match  
    
   ?                 a                aa  
                     b  
    
   a?b               a2b              ab  
                     axb              acesdb  
                     a\_b  
    
   ?txt              atxt             notes.txt  
                     3txt             txt  
                     qtxt
* **The Character-Class :** A character-class is enclosed in square brackets [ ] and may include a list of characters or a range of characters separated by a dash. The character class matches any one character in the list or ranges. Or의 개념   
    
  Ambiguous Pathname       Matches        Does not match  
    
   [abc]                    a                A  
                            b                aligator  
                            c                aaaaa  
    
   test[12]                 test1            test12  
                            test2            test3  
    
   notes[127-9]             notes1           notes3  
                            notes2           notes  
                            notes7           notes.txt  
                            notes8           notes127

\*\* [**12-5]** : 1,2,3,4,5 포함함/ **-**의 의미는 그사이에 있는 숫자는 가능

* You can invert the meaning of a character class by placing the symbol **'!'** on  Phobos or **'^'** Matrix at the start of the character-class.  That will make it match any of the characters which are not in the character-class.  
  \*\* ! : means not

Ambiguous Pathname     Matches        Does not match  
  
 [!a-c]\*               drift           aligator  
                       4234run         bowl  
                                          cat

aaligator  
  
 a[!b]c                a2c             abc  
                       alc             allc : 11은 2개라서 안됨

명령어는 1개만 필요

* Example of filename expansion

let’s assume the following regular files are contained in our current directory:

work1.txt work2.txt work3.txt work4.c worka.txt working.txt

Note the results from using filename expansion

ls work**\***

work1.txt work2.txt work3.txt work4.c

worka.txt working.txt

ls work**?.txt**

work1.txt work2.txt work3.txt worka.txt

ls work**[1-3].txt**

work1.txt work2.txt work3.txt

ls work**[!1-3].txt**

worka.txt

* **Globbing** : The process of converting an ambiguous filename into a list of matching file names is called 'globbing' and is performed by the shell. On other operating systems, the equivalent operation is performed by the command and not by the shell.

**\*\*shell :** Shell은 사용자와 커널이 상호작용 할 수 있도록 도와주는 유틸리티로 사용자가 입력한 명령어를 커널이 이해할 수 있도록 변환해주는 역할을 한다.예를 들어, shell에서 ls 라고 입력하고 엔터를 치면 파일 리스트가 화면에 보여지게 되는데 여기서 사용자가 입력한 ls가 "명령어"가 되고 shell에 의해서 커널이 인식할 수 있는 형태로 변환되어 커널은 "아~ 파일리스트 보여달라는구나" 라 인지하고 화면에 파일리스트를 표시하게 된다.

* Unix shell

1. Command interpreter for UNIX
2. Acts as a mediator between user and UNIX kernel
3. Processes and/or executes user commands
4. More than one command can be executed on one command line when separated by a semi-colon
5. You will be learning approx. 30 Unix commands in this course : This is a small, compared to the the 1000+ Unix commands out there
6. The term command and utility mean the same in Unix
7. There are several kinds of shells available for UNIX

Most popular shells are:

– C shell (this is not the C programming language)

– Korn shell – used with Unix

– Linux machines most often use the BASH shell (Bourne-Again Shell)

1. Each user on one machine can run a different shell
2. UNIX scripting = UNIX shell programming
3. Shell basics

command history

<Up> or <Down> : move to previous command or next command

fc -l : display last 16 commands

history : display all commands in buffer

!num : re-execute command number "num"

!xxx : re-execute last command beginning with string "xxx"

**\*\*globbing :** 모든 shell은 파일 시스템에서 특정한 이름패턴을 만족하는 파일들을 선택할수 있는 대표문자기능을 지원함. 명령중에서 적어도 하나의 대표 메타문자를 포함하는 단어는 패턴처럼 처리되고 일치하는 모든 파일 이름의 알파벳 순으로 정렬된 목록으로 대치됨. 이러한 패턴 치환동작을 globbing 이라함

* Be aware that because globbing is performed by the shell, globbing is applied to all arguments, whether they are supposed to represent files or some other information.  
    
  [e.g] the 'echo' command displays information on the screen.  If you type 'echo \*', the asterisk will be replaced by a list of all of the files in the current directory. To avoid this, use double-quotes around the argument.  
    
  In order to demonstrate the use of 'wildcard' characters, issue a command to create the following empty files in your current directory:  
    
  Try 'touch a1 a11 a123 a1234 a23 a22 a3 1 11 123'  
    
  Now issue a command to provide a compact listing of all files that begin with the letter 'a'.   
    
  $  ls a\*  
  a1  a11  a123  a1234  a22  a23  a3  
    
  Now, issue a command to provide a compact listing of files in your current directory that begin with 'a' and end with '3'  
    
  $  ls a\*3  
  a123  a23  a3  
    
  To demonstrate, the difference between \* and ? used as wildcards, issue the command 'ls a?3'  
    
  $  ls a?3  
  a23   
    
  Notice that using the '\*' symbol matches zero or more characters, as opposed to matching exactly one character with the '?' symbol. You can also use multiple '?' symbols to match a certain number of characters.  
    
  $  ls a??3  
  a123  
    
  Character classesbehave likethe '?' wildcard, except you can specify what characters '?' can or cannot represent.  
    
  First, let's provide a compact listing for files that begin with either the lowercase letter 'a' (case matters) or begins with the number '1'.  
    
  $ ls [a1]\*  
  1  11  123  a1  a11  a123  a1234  a22  a23  a3  
    
  Now let's provide a compact listing for files that end with the number '1'  
  or '2' or '3', using the range '[1-3]'  
     
  You can also combine ranges and/or lists.  For example, to list all files that begin with a letter (whether upper or lowercase), use the ambiguous filename

[a-z A-Z]\*  
  
Now let's invert a character class using '!'.  Let's display a compact listing of all files that do not start with 'a'.  
ls [!a]\*

json13@matrix:~> cd uli101nw17

json13@matrix:~/uli101nw17> cd assignment

json13@matrix:~/uli101nw17/assignment> ls

1a @asd a1 a123 a2 a3 a4 abc

json13@matrix:~/uli101nw17/assignment> touch a1 a11 a123 a1234 a23 a22 a3 1 1 123

json13@matrix:~/uli101nw17/assignment> ls

1 123 1a @asd a1 a11 a123 a1234 a2 a22 a23 a3 a4 abc

json13@matrix:~/uli101nw17/assignment> ls a\*

a1 a11 a123 a1234 a2 a22 a23 a3 a4 abc

json13@matrix:~/uli101nw17/assignment> ls a\*3

a123 a23 a3

json13@matrix:~/uli101nw17/assignment> ls a?3

a23

json13@matrix:~/uli101nw17/assignment> ls a??3

a123

json13@matrix:~/uli101nw17/assignment> ls [a1]\*

1 123 1a a1 a11 a123 a1234 a2 a22 a23 a3 a4 abc

json13@matrix:~/uli101nw17/assignment> touch 11

json13@matrix:~/uli101nw17/assignment> ls

1 11 123 1a @asd a1 a11 a123 a1234 a2 a22 a23 a3 a4 abc

json13@matrix:~/uli101nw17/assignment> ls [al]\*

a1 a11 a123 a1234 a2 a22 a23 a3 a4 abc

json13@matrix:~/uli101nw17/assignment> ls [!a]\*

* 1. 11 123 1a @asd

1. Directory Management Commands : Let's work with pathnames and directories.

* You create directories by issuing the linux command:

mkdir <directory-name>

* The command above is using a relative pathname; therefore, the directory will be created as a child directory. Below is an example of using an absolute pathname to create a directory:  
    
  mkdir /students/username/directory-name
* To save time, you can create more than one child directory under the current directory by placing multiple directory names within same Linux command  
    
  mkdir d1 d2 d3
* planning Directories : good directory organization requires planning
* Group information together logically
* Plan for the future : used dated directories where appropriate

[e.g.] ~/Christmas/2001 /chrismas/2002

* Too few directories=excessive number of files in each, too many directories=long pathnames
* Let's verify the file by entering  
    
  ls d\*
* You can also issue a Linux command to create directories even if their 'parent' directories do NOT exist. To do this, add the '-p' (parent) argument to the mkdir command.  
    
  Let's create a child directory with its parents. ( at the same time)  
    
  mkdir -p  testing/1/2/3

Testing is 1’s parents , 1 is 2’s parents, 2 is 3’s parents  
Now we will issue a command to verify that the directories have been made, using a relative pathname.

ls -l -d  testing /1/2/3  
  
Now issue the same command with an absolute pathname.  (Note that your current directory is your home directory:  
/home/ayesha.manzer ).

ls  -l -d  /home/ayesha.manzer/testing/1/2/3      
  
You can change your current directory using the 'cd' command.  Change to the '/tmp' directory.  
  
 cd /tmp  
  
We can check our current directory with the 'pwd' command (print working directory).  
  
  pwd  
 /tmp  
  
Now use a relative-to-home pathname to list the contents of the directories that you created in your home directory a few minutes ago.  
  
ls -l -d ~/testing/1/2/3                                    
  
If you do not specify any arguments for the 'cd' command, you will set your current directory back to your home directory.  
  
pwd  
  
Create the empty file called 'file1' in the directory labeled '3' that you just created. You may use any of the types of pathnames.  
  
$ touch /home/ayesha.manzer/testing/1/2/3/file1  
  
Now issue a command to verify that you created the file.

Review Section\_week2  
  
The purpose of this section is to reinforce your skills with issuing UNIX commands covered in the previous section.  
  
The answers you complete in this section will be e-mailed to you and your professor (and also saved in case of problems with the e-mail service).  You will be asked for confirmation before the results are sent. No mail will be sent if you exit from this section before reaching the end.  
  
You will be asked to enter a single UNIX command for the following situations. If you are unable to successfully issue the UNIX command after everal attempts, a hint will be provided.  
  
**Question 1**  
Enter the Linux command to create both a parent directory called 'unx122\_lab3' and it's child directory called 'work' at the same time. Assume that directory 'unx122\_lab3' will branch-off your home directory and that you are in your home directory to start. Use a relative pathname.

mkdir(space) -p unx122\_lab3/work

**Question  2**  
Enter a Linux command to create the empty file called 'labtest1' in the directory 'unx122\_lab3' (you are still in your home directory).

Touch unx122\_lab3/labtest1

Touch /home/json13/unx122\_lab3/labtest1

Touch ~/unx122\_lab3/labtest1  
  
**Question 3**  
Assume that your current directory contains the files 'labtest', 'labtest1', 'labtest2', 'labtest2a', 'labtest3', and 'labtest4'.  
Issue a command to delete only the files 'labtest1' and 'labtest2' using one ambiguous pathname.  
done  
if I want to remove the only test2, test3

test 1 test 2 test3 test4

rm labtest[12]

**Question 4**  
Assume that you are NOT currently in your home directory.  Enter a command to copy all files in your home directory beginning with the letter 'a' to the current directory.

cp ~/a\* .

**Question 5**  
Enter a command to delete all files that have filenames starting with 'labtest', except 'labtest' itself (Delete all files starting with 'labtest'  
followed by one or more characters).

rm labtest?\*

labtest1 latest2   
  
**Question 6**  
Enter a Linux command to remove a directory called 'assignments' and all of it's contents, and allow the system to prompt the user to remove each file or directory.  
Assume you are in the home directory, which happens to be the parent directory of 'assignments' directory.

rm -ir assignments

**Question 7**  
Here are two inverted-tree diagrams. Issue a command to change the left diagram to the right diagram.  Assume that you are in your home directory and use relative pathnames. [home] is your home directory:  
  
         [home]                       [home]  
           |                            |  
     +-------+-------+           +-------------+  
     |       |       |           |             |  
    unx122   |    courses       ideas       courses  
     |       |                                 |  
     |      ideas                            unx122  
    notes                                      |  
                                             notes  
mv unx122 courses

**Question 8**  
Issue a command to delete all files in your current directory with 2-character names.

rm ??

**Question 9**  
Issue a command to delete the directory 'courses' (in your home directory) and all of its children.  
Use an absolute pathname (remember that your home directory is /home/username ).

rm -r /home/username/course

**Question 10**Enter a command to make the root directory your current directory.

cd /