CSCI 211
UNIX Lab

Basic Unix Commands (3)

Dr. Jiang Li

Jiang Li, Ph.D.
Department of Computer Science

Today’s Focus

- Environment Variables
- Who login the system
- Wildcard characters
- Search in directories and files

Environment Variables

- A set of values affecting the behavior of processes
- Examples
  - HOME: the home directory path
  - PATH: the paths to search for commands
  - USER, LOGNAME: the current logged in user
  - SHELL: the shell that the user are using
- Show value

  e.g. `echo $HOME` *(Put a dollar sign before variable name)*
- Set value

  e.g. `export PATH=~/bin:/usr/bin:$PATH`

$PATH

- Try the following

  ```
  cp /lab3/hi .
  hi
  ```

- hi is not in $PATH!
- Solution 1
  - Refer to hi with its path

    ```
    ./hi
    ```

- Solution 2
  - Add current directory $PATH

    ```
    Export PATH=.:$PATH
    ```

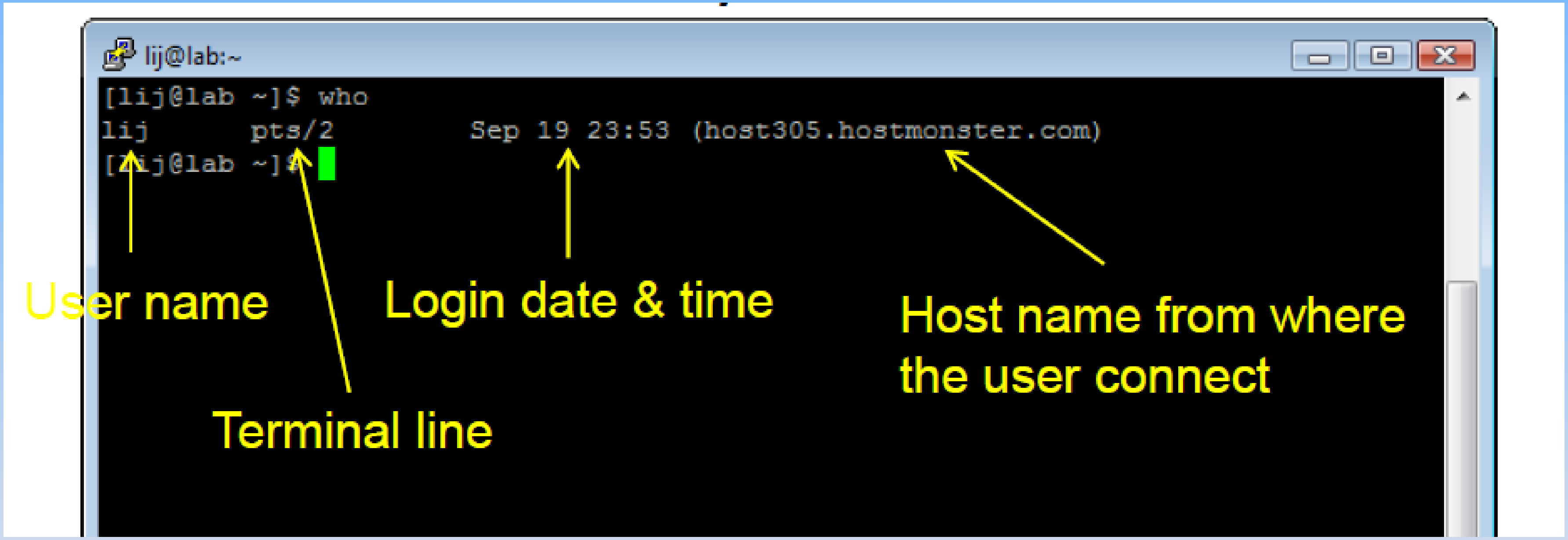
  - Verify

    ```
    Echo $PATH
    ```

  - Try hi again

The `who`, `finger` command

- `who` command shows who are login the system at the same time as you are.



```
[lij@lab ~]$ who
lij      pts/2        Sep 19 23:53 (host305.hostmonster.com)
[lij@lab ~]$
```

- `finger` command have similar functionality as `who`, while shows more detailed information than `who`.

Wildcard Character

- Wildcard
  - A character that may be substituted for other characters
  - * matches zero or more characters
  - ? matches exactly one character

Wildcard Character Examples

- List the files with a name starting with "test"
  ```
  ls test*
  ```
- List the files with a 5-character name starting with "test"
  ```
  ls test?
  ```
- List the files with a 5-character name ending with "test"
  ```
  ls ?test
  ```
- List the files whose name has the "t" as the 3rd character and "s" as the 6th character
  ```
  ls ??t??s*
  ```

Multiple Commands in One Line

- Multiple commands can be executed by one line by separating the commands with ;
- Example

```
mkdir temp; cp filename temp/; cd temp
```

Search in Directories - find

- Search file/directory names under a path
- Syntax:

```
find [flags] [path] [expression]
```

- Common usage:

```
find path -name pattern -print
```

- Example:

```
find /home -name "test" -print
find / -name "test*" -print
```

Search in Files - `grep`

- Search file's content, output the lines with match(es)
- Syntax:

```
grep [options] PATTERN [FILE/DIR...]
```

`-i:` ignore case of alphabetic characters

`-E:` use regular expressions for search

`-n:` shows the line number of matched content

`-r:` search all files and subdirectories under a given directory

Search in Files - `grep`

- Examples

  `grep test check` (search the file ‘check’ for ‘test’, case sensitive)

  `grep test check/` (search all files under the check directory for ‘test ’, case sensitive)

  `grep -i test *` *(search all files containing ‘test’ in current directory, ignore case)*

  `grep -r test *` *(search all files and including subdirectories containing ‘test’ from current directory .)*

  `grep -n test license` *(search file `license`, output the line number that containing ‘test’)*

  `grep -E "[0-9]+" license` *(search file `license`, output the lines that containing a number)*

Some Reg. Exp. Examples in grep

```
grep -E "ab(cd|ef)g" filename
grep -E "abc?d" filename
grep -E "abc*d" filename
grep -E "abc+d" filename
grep -E "ab[cdef]g" filename
grep -E "ab[c-f]g" filename
grep -E "ab[a-zA-Z0-9]g" filename
grep -E "ab[a-zA-Z0-9]*g" filename
```