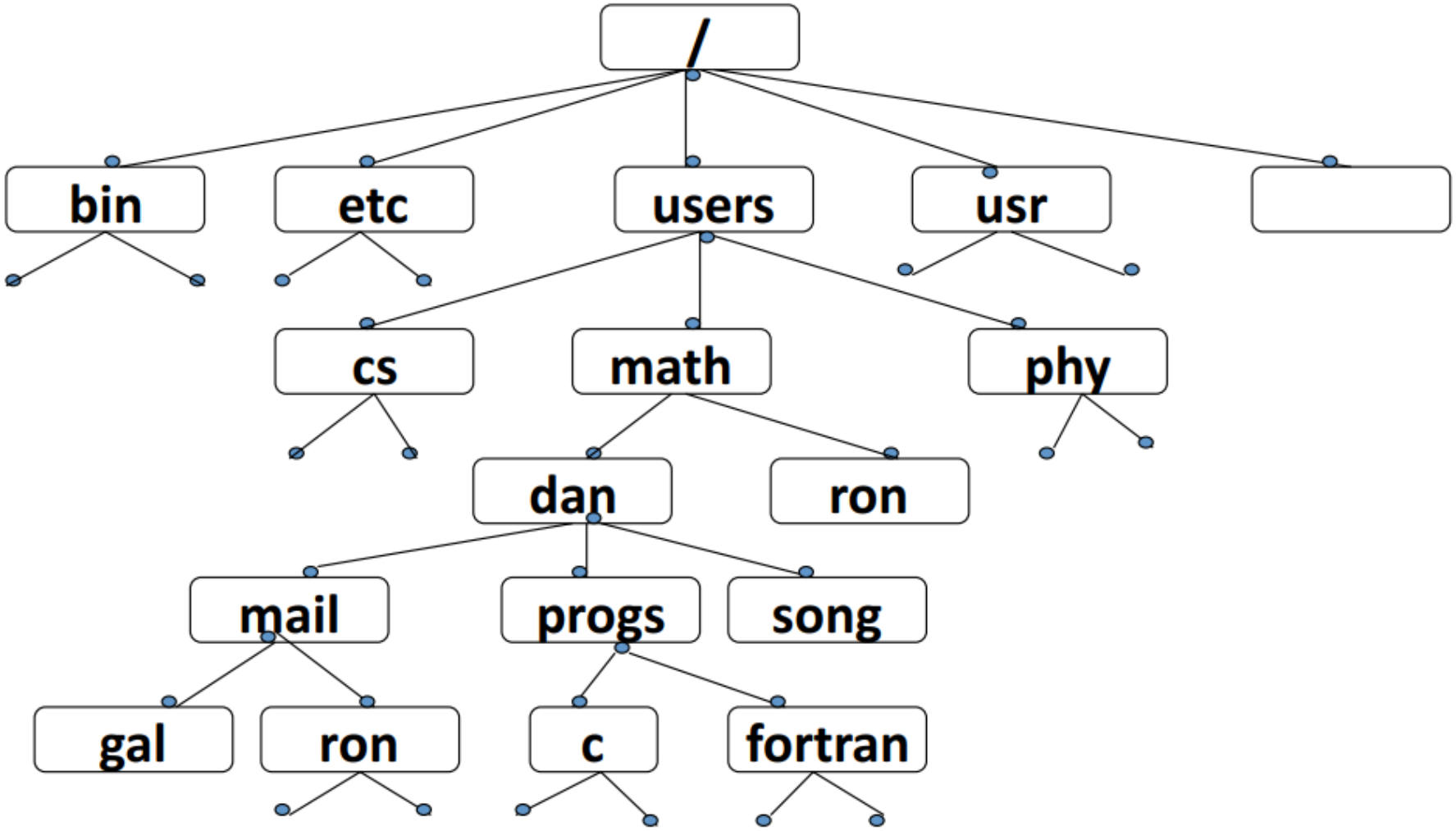

Linux™
#2

The Unix shell

- . . . is your interface with the UNIX system, the middleman between you and the kernel.
- . . . is an program so called 'interpreter'. It operates in a simple loop:
  - It accepts a command
  - it interprets the command
  - it executes the command
  - and then waits for another command.
- . . . displays a "prompt," to notify you that it is ready to accept your command: pwd

Hierarchical File System (Tree Structure)



The Unix File System

- . . . is hierarchical (resembling a tree structure).
- The tree is anchored at a place called the root, designated by a slash ”/”.
- Every item in the file system tree is either a file, or a directory.
- The path is the location of a file or directory within the tree, e.g. /usr/sbin/bzip2, ../../erick
- . . . support access control by defining permissions for read, write, and execution.

How to modify my path?

- If you just want to add a directory to your path on the command line, you can use:

```
$PATH="$PATH:/tmp"; export PATH
```

The Unix File system, cont'd

- **read**: allows you to read a file or to list the directory's contents
- **write**: allows you to modify the contents of a file or to alter the contents of the directory, i.e., to add or delete files.
- **execute**: allows you to run the file, if it is an executable program, or script or to change into this directory.

ls -l

FS Linux commands

- **man**: display the on-line manual pages
- **apropos**: search the a database for strings of system commands
- **ls**: list directory contents
- **mv**, **rm**, **cp**: move, remove, and copy files
- **mkdir**, **rmdir**: make and remove directories
- **file**: determine file type
- **chmod**, **chown**: change file access permissions, and file owner or group

Other important commands

```
find /etc/ -type f  -name "pass*" | xargs grep bash | wc -l
```

- **find**: powerful search for files in a directory hierarchy
- **cat**: concatenate files and print on the standard output
- **less**, **more**: paging through text at monitor
- **head**, **tail**: show the first and last lines of a file
- **sort**: sort lines of text files
- **grep**: print lines matching a pattern

Other important commands

- **wc**: print the number of newlines, words, and bytes in files
- **xargs**: build and execute command lines from standard input

Input and output redirection

- **stdin**: (0) the input of the program, usually the keyboard
- **stdout**: (1) where the output of the program goes
- **stderr**: (2) where the error messages usually go
- >, <, 2 > : redirects stdout, stdin, stderr
- 2 > &1 : redirects stderr to stdout
- A |B : redirects stdout of 'A' to the stdin of the 'B'

<: mail –s "news today" abc@gmail.com < newFile
2>: telnet localhost 2> errorfile

Wildcards in the bash

- ? : a single character
- * : zero or more characters
- [b -d] : either b, c, or d
- {conf, loc} : either ’conf’ or ’loc’

```
user% ls -d /etc/*[b-h ]?.{ conf ,loc }
```

Quoting in the bash

- 3 characters for quoting:
  - single quote (’),
  - double quote (”)
  - Backslash

Quoting: the differences

- A **backslash** protects the next character, except it is a newline
- **Single quotes** ('...') protect everything (even back slashes, newlines, etc.) except single quotes, until the next single quote.
- **Double quotes** ("...") protect everything except double quotes, backslashes, dollar signs, and backquotes, until the next double quote.
- A **backslash** can be used to protect ", $, or ' within double quotes. A backslash that does not precede ", $, ', or newline is taken literally.

Processes and system info

- **ps** reports process status
- **kill** terminates a process (sends a signal)
- **df** reports file system space usage
- **du** estimates space usage
- **quota** display space and time limits
- **date** prints or sets system date and time
- **time** times a command, gives resource usage

File processing

- grep prints lines matching a pattern
- find searches for files in hierarchy
- diff find differences between two text files.
- cut removes sections from each line of a file.
- awk pattern scanning and processing language.
- sed stream editor
- sort sorts lines of text files
- uniq removes duplicates lines from file.
- tr translates or deletes characters.

```
$sed 's/unix/linux/' file.txt
$ awk '{print}' file
$ cat  file | tr “[a-z]” “[A-Z]”
$ cat  file | tr “[:lower:]” “[:upper:]”
```

Regular expression

- These are used in **grep**, **awk**, **sed** and other programs that use the *regex* library.
  - Here a **sed** command is used to perform operations on text, e.g.

```
$ echo "abcdefg" | sed 's/abc/zzz/g‘
zzzefg
```

regexp metacharacters

- . Matches any single character
- [...] matches any of the characters in the brackets.
- [a-z] matches any of the characters in the given range.
- [^...]matches any character not in the given set.
- ? Matches the preceding element 0 or 1 times. So “ab?” matches “ a” or “ ab”.
- * matches the preceding element 1 or more times.
- + matches the preceding element 1 or more times.
- {n} matches the preceding element exactly n times.
- {n,m} matches the preceding element at least n, but not more than m times.

ביטוים רגולריים בשמות קבצים

- לכל תו רגיל

```
ls –l /usr/include/stdio.h
```

- כוכבית – * : מתאים לרצף של תווים כלשהם.

```
ls –l /usr/include/*.h
grep –i erick * */*
```

- סימן שאלה – ? : מתאים לכל תו בודד.

```
ls –l /usr/include/*.?
grep  fredj *
```

- סוגריים מסולסלים – {str1, str2, str3}

```
cp / usr / include /{stdio, stdlib}.h /tmp
```

- (tilda) ~ – ה home directory של המשתמש

```
cp ~moshe/ .cshrc ~/
```

ביטוים רגולריים ב- grep

- כל תו שאינו תו מיוחד מתאים לעצמו.

```
grep erick *
```

- הסימן ˆ בתחילת ביטוי מצמיד את הביטוי לתחילת השורה. הסימן $ בסוף שורה מצמיד את הביטוי לסוף שורה.

```
grep ^erick *
grep  moshe$ *
grep ^rachel$ *
grep ^$ *
```

- "." (נקודה) מתאימה לכל תו פרט לתו \(newline n)

```
grep . *
```

יחפש את כל השורות שאינן ריקות (יש בהן תו כלשהו).

```
grep .avid *
```

ימצא את השורות בהן מופיע "david" או "ravid" וכו.

ביטוים רגולריים ב- grep

- רצף סימנים בסוגריים מרובעים [ ] מתאימים לתו אחד מתוכם. אם הסימן ^ מופיע ראשון בתוך הסוגריים המרובעים התבנית מתאימה לכל תו פרט לשאר התווים בתוך ה- [ ]. הסימן "-" (מינוס) בין שני תווים מציין רצף תווים בין שני התווים שמלצדדיו.

grep [dr]avid *

יחפש את השורות שבהן מופיע "david" או "ravid" אך לא "savid"

grep [1-9][0-9]4 *

יחפש את השורות שבהן מופיעים רצפים של שלוש ספרות כאשר הראשונה אינה אפס והאחרונה היא 4.

grep ^[^a-zA-Z] *

יחפש את כל השורות שלא מתחילות באות אנגלית (קטנה או גדולה).

grep -ב

- אחרי כל אחד מהנ"ל יכולה לבוא הכוכבית "*" וביטוי החדש מתאים לרצף של אפס או יותר מופעים של מה שבא לפני הכוכבית. יש לשים את הביטוי במרכאות כדי לא לבלבל את ה- s .

grep "da*d" *

ימצא את השורות שבהן מופיע "daaaaad" , אבל לא "dadad" כי הכוכבית פועלת רק על בר האחרון שהיא רואה וזה "a"

- אם אחרי כל אחד מהביטויים הקודמים באים סוגריים מסוסלים עם מספר בתוכם, הביטוי החדש יחפש את מה שיש לפני הסוגריים המסולסלים מספר פעמים כמו שרשום בתוך הסוגריים , המסולסלים. צריך להקדים את הסוגריים עם backslash כפי שמופיע בדוגמאות למטה.

grep "me\{2\}t" *

ימצא את " meet" אבל לא את "met"

grep "^m[aouei]\{2\}" *

ימצא את כל השורות המתחילות באות "m" ואחריה שתי תנועות

grep -ב

- סימנים מיוחדים "\>" ו- "\<" מתאימים לתחילת מילה ולסיומה בהתאמה.

```
grep "\<[a-zA-Z]\{4\}\> *
```

ימצא את כל השורות שיש בהן מילים בנות 4 אותיות