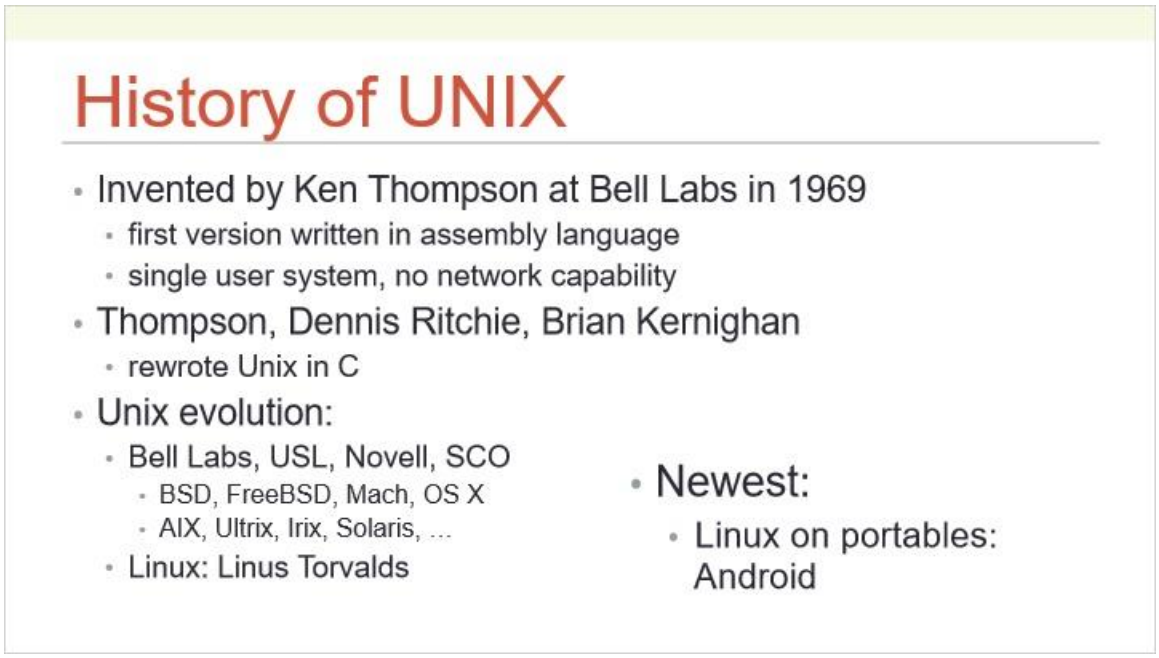
Midterm Review

1. Midterm Review

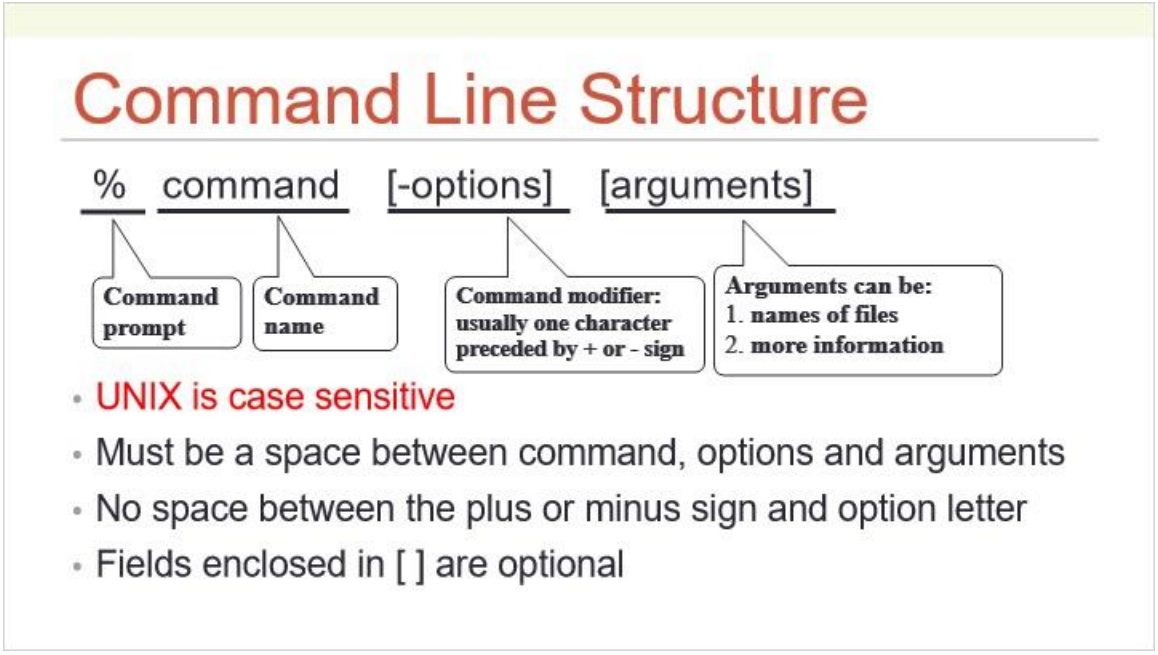
1.1 Overview

1.2 Topics covered so far

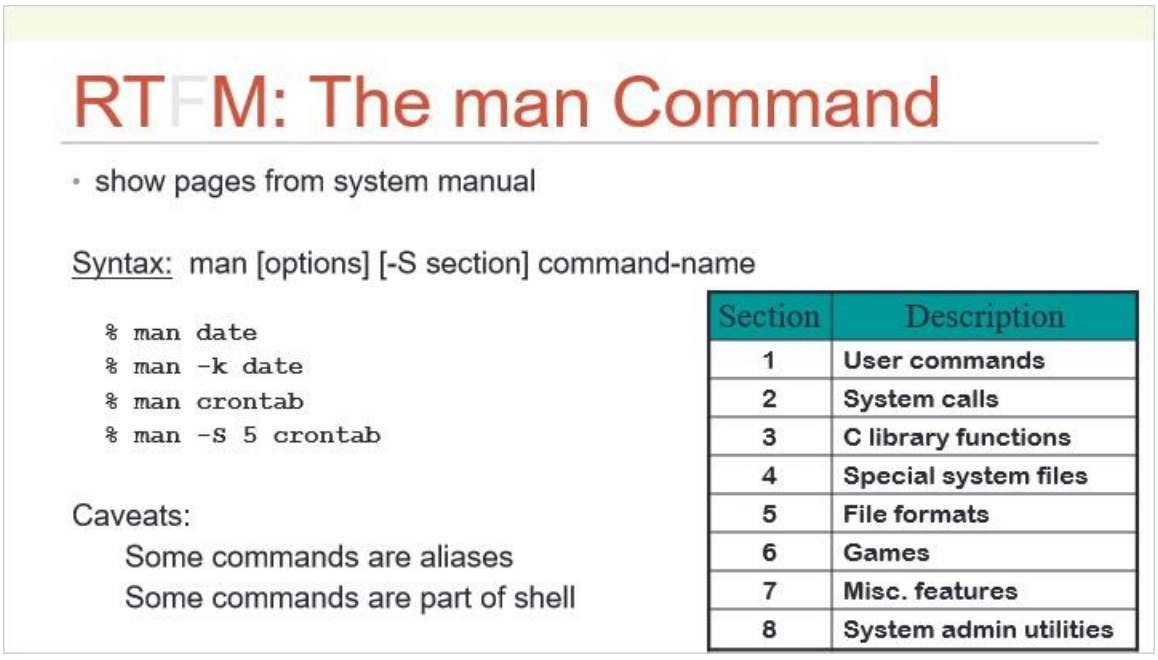
1.3 History of UNIX



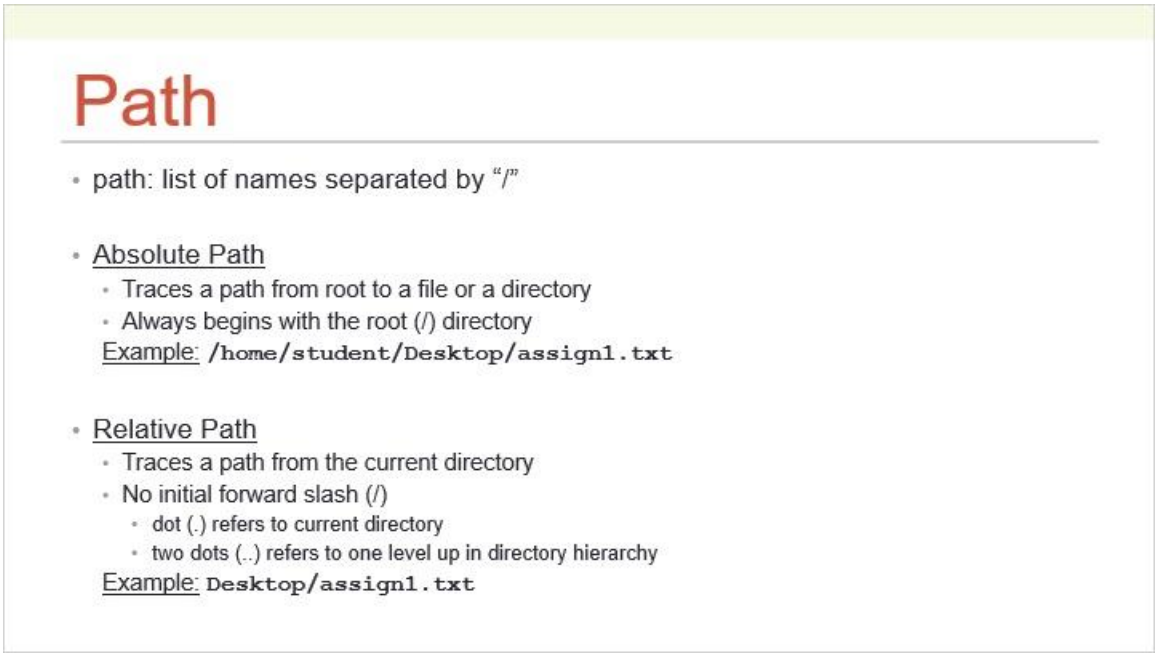
1.4 Command Line Structure



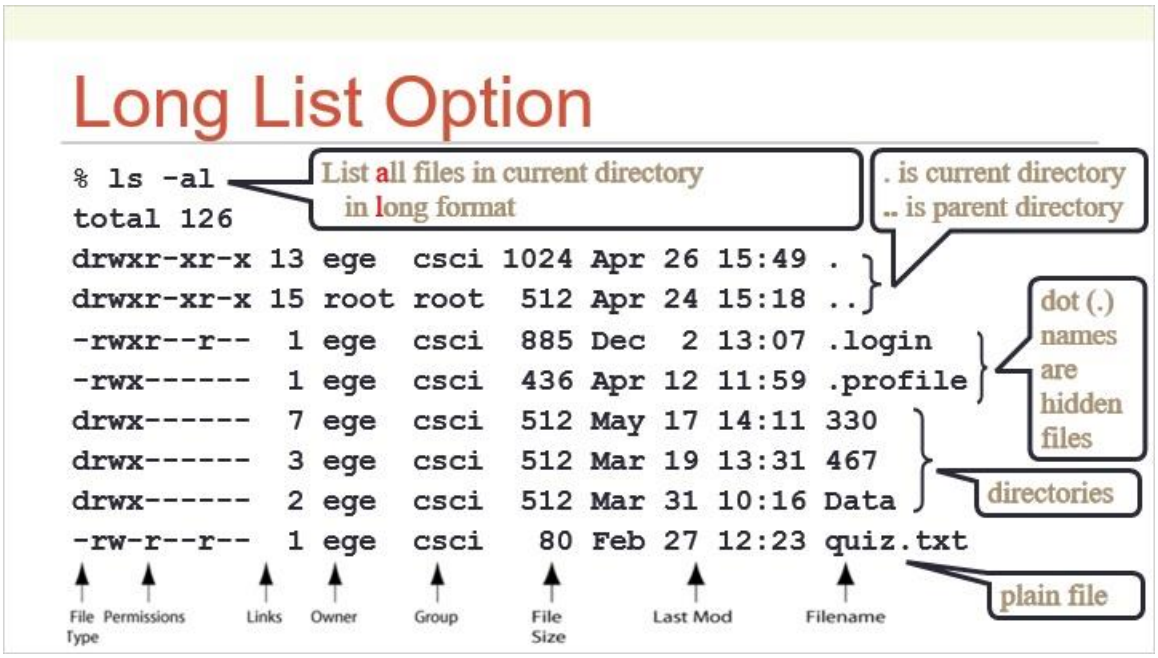
1.5 RTFM: The man Command



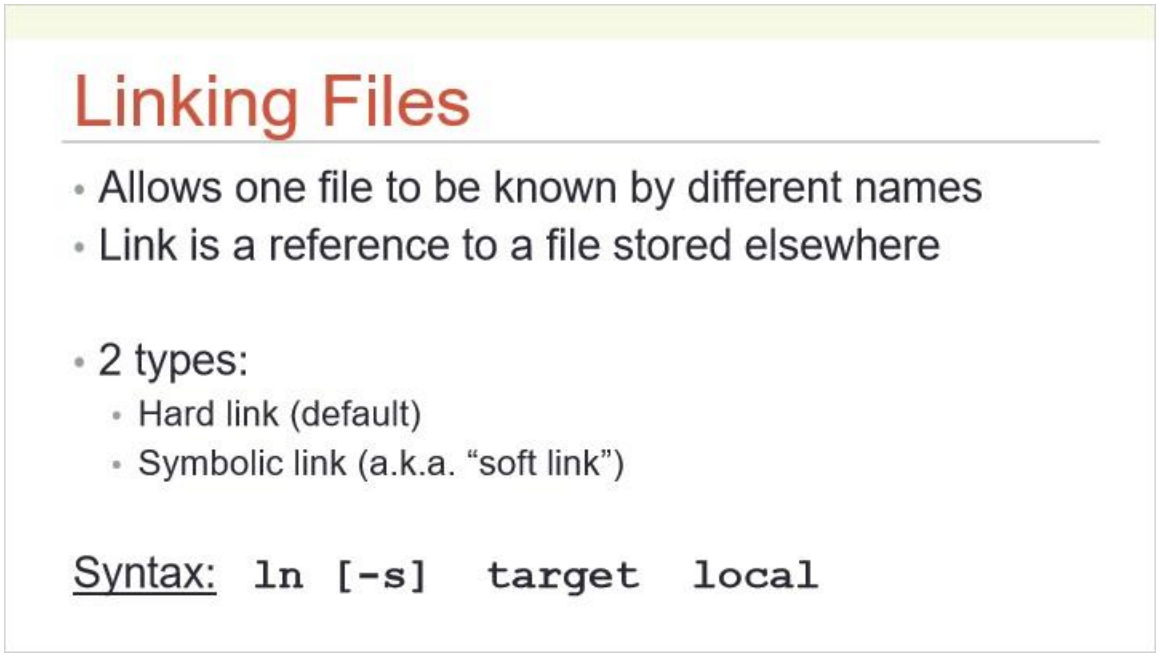
1.6 Path



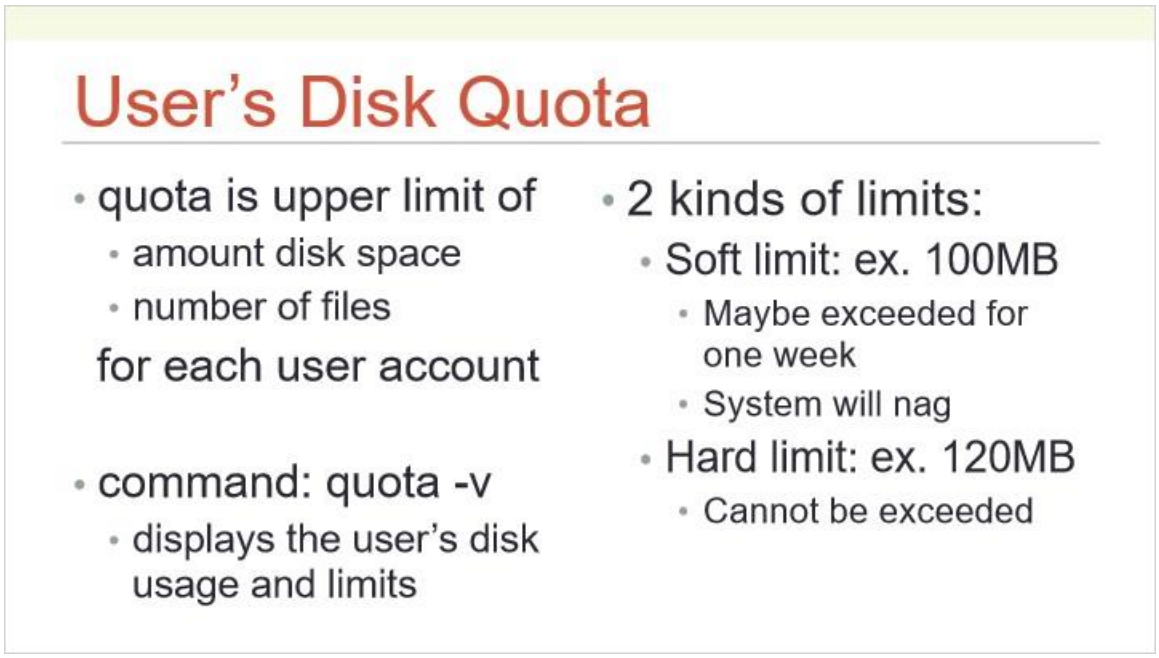
1.7 Long List Option



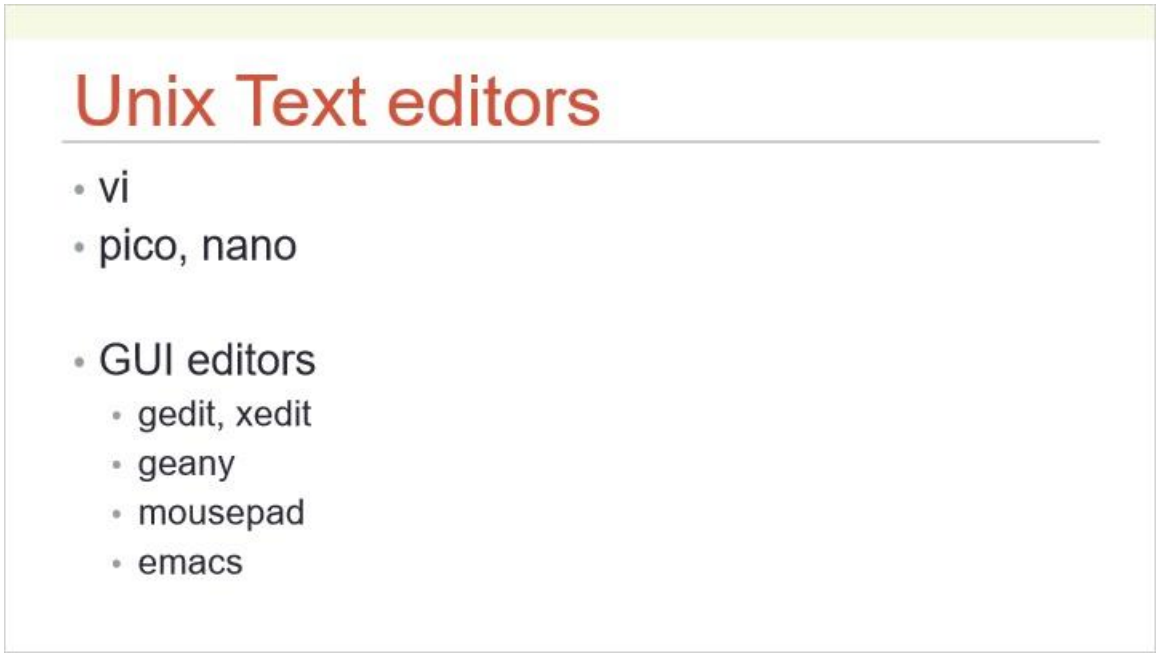
1.8 Linking Files



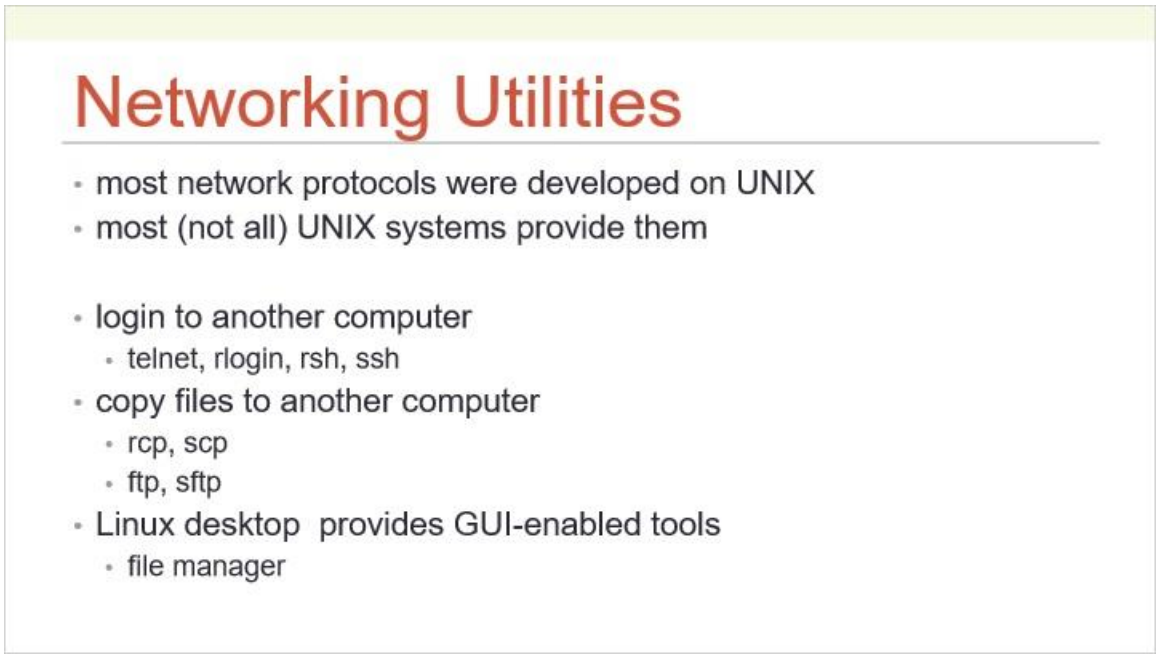
1.9 User's Disk Quota



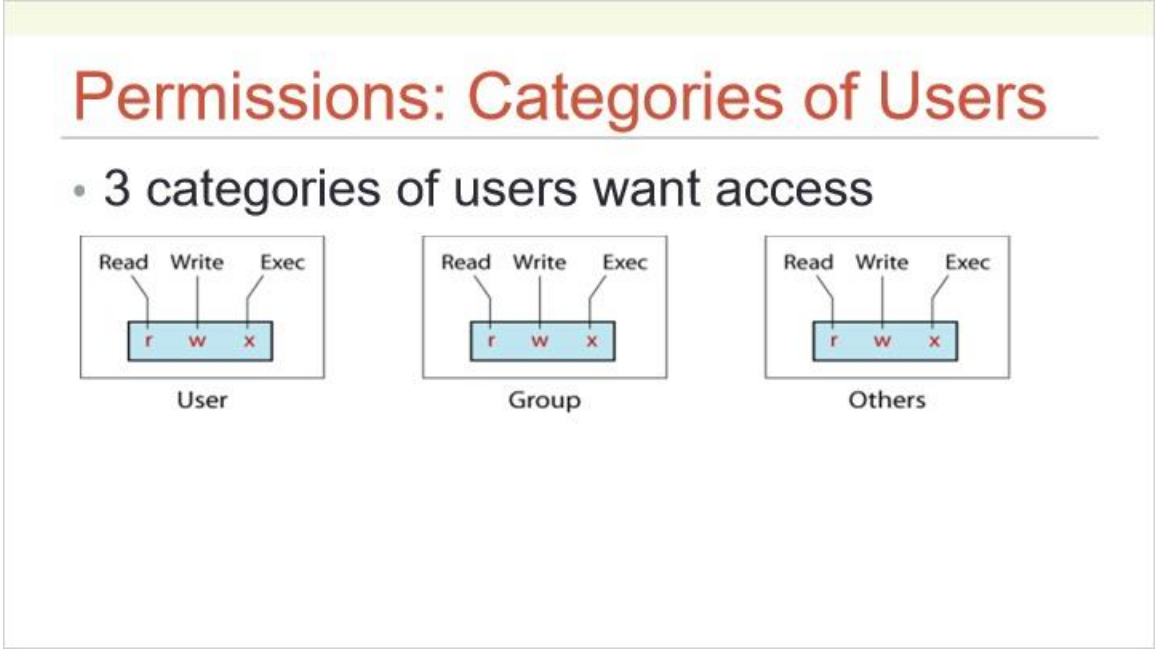
1.10 Unix Text editors



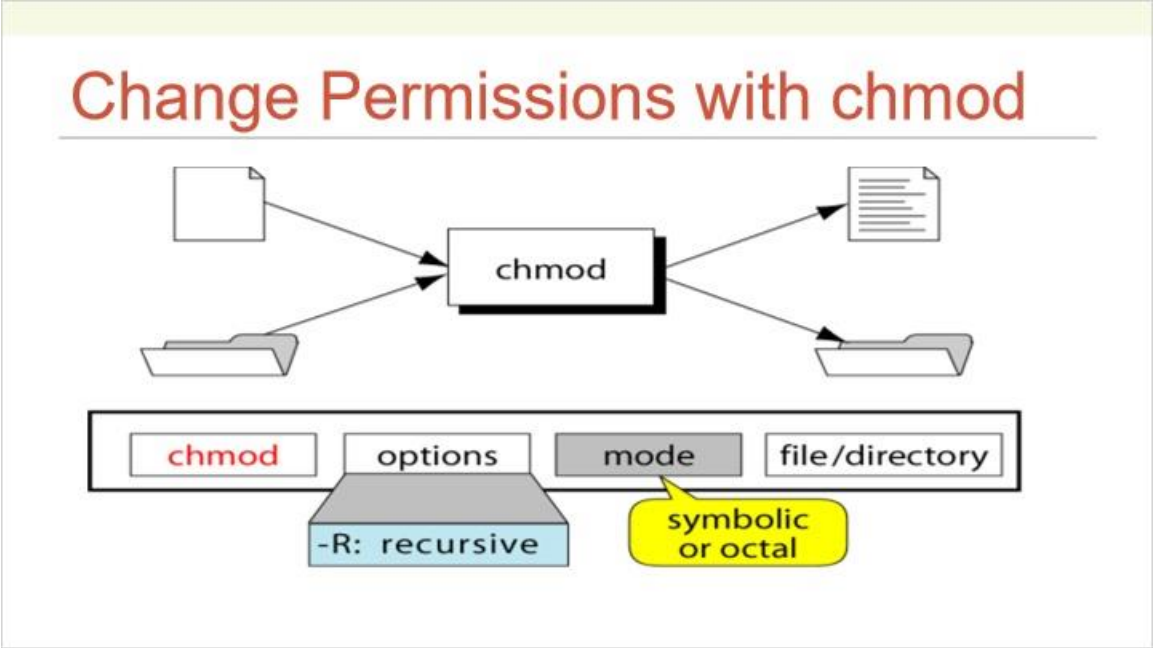
1.11 Networking Utilities



1.12 Permissions: Categories of Users

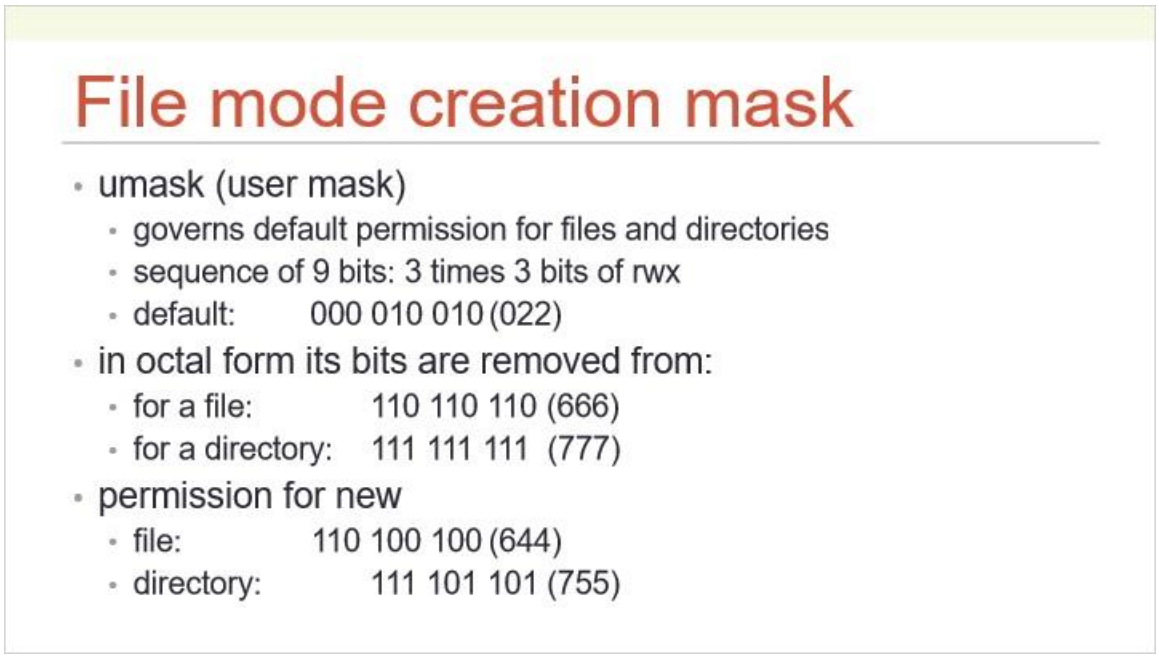


1.13 Change Permissions with chmod

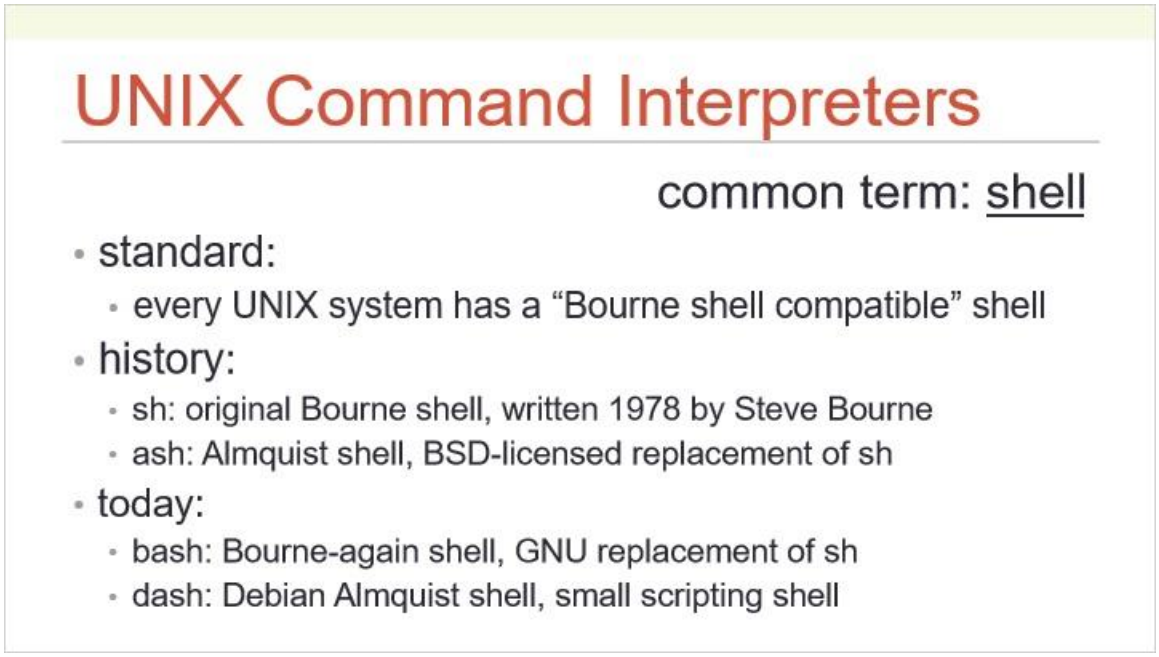


1.14 Special Permissions

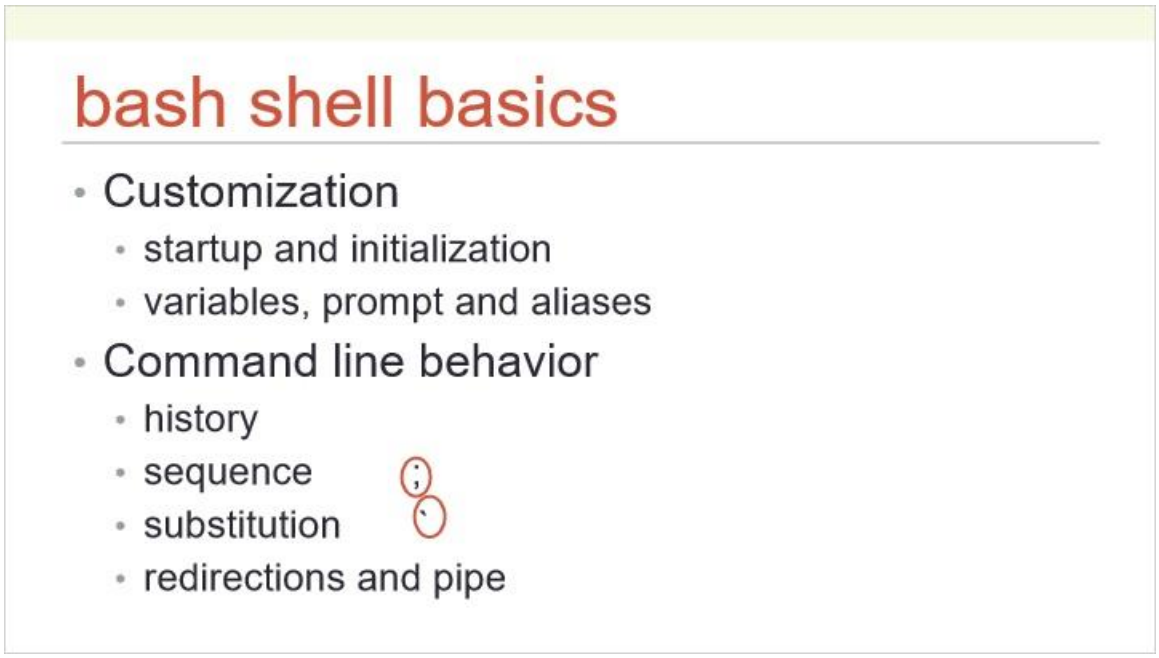
1.15 File mode creation mask



1.16 UNIX Command Interpreters



1.17 bash shell basics



1.18 Summary: Redirections and Pipe

Summary: Redirections and Pipe

Command Syntax	Meaning
command < file	redirect input from *file* to *command*
command > file	redirect output from *command* to *file*
command >> file	redirect output of *command* and appends it to *file*
command > file 2>&1 command &> file	add error output to standard output, redirect both into *file*
command1 | command2	take/pipe output of *command1* as input to *command2*
command << LABEL	take input from current source until *LABEL* line

*1.19 Wildcards: * ? [ ] { }*

Wildcards: * ? [ ] { }

A pattern of special characters used to match file names on the command line

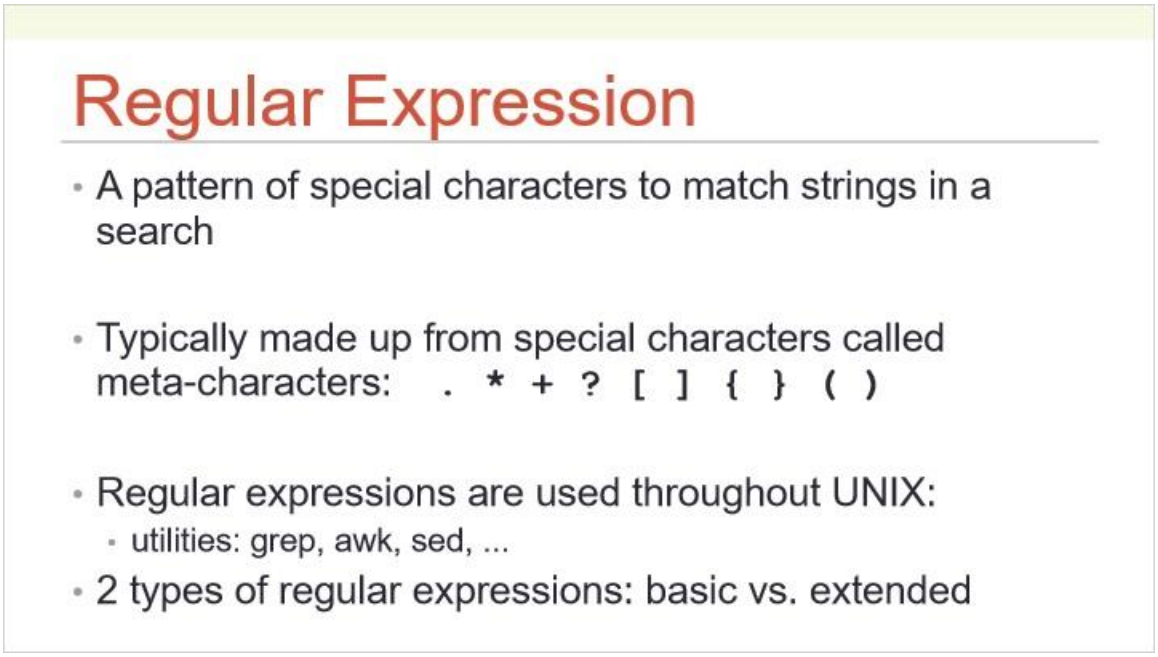
* zero or more characters

```
% rm *
% ls *.txt
% wc -l assign1.*
% cp a*.txt docs
```

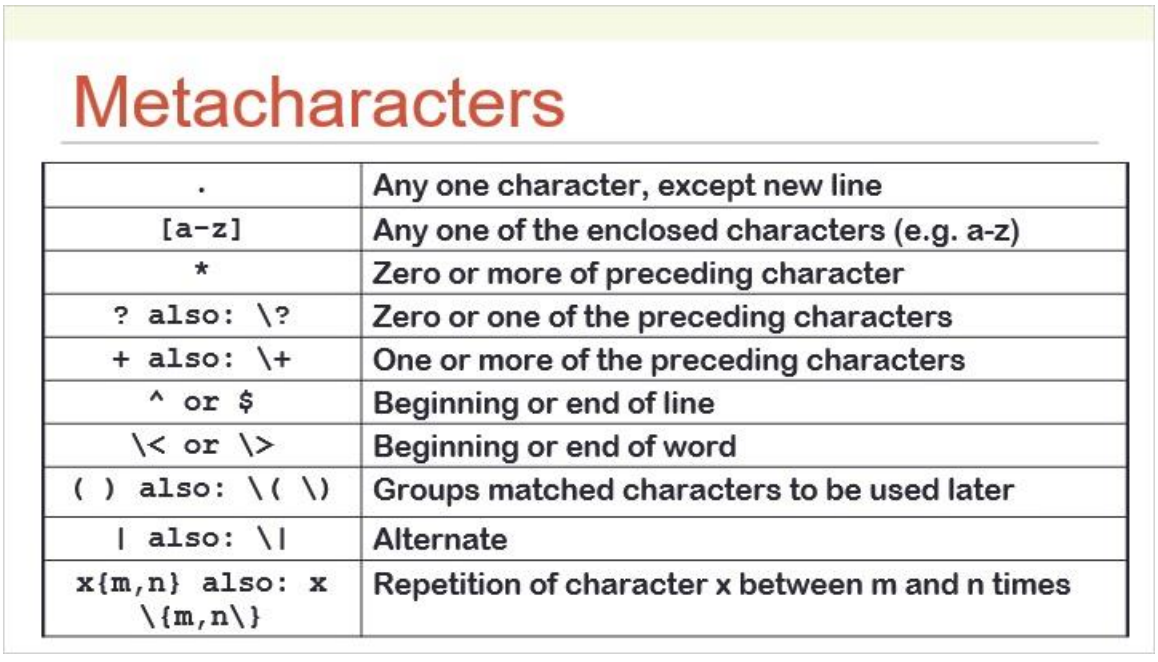
? exactly one character

```
% ls assign?.cc
% wc assign?.??
% rm junk.???
```

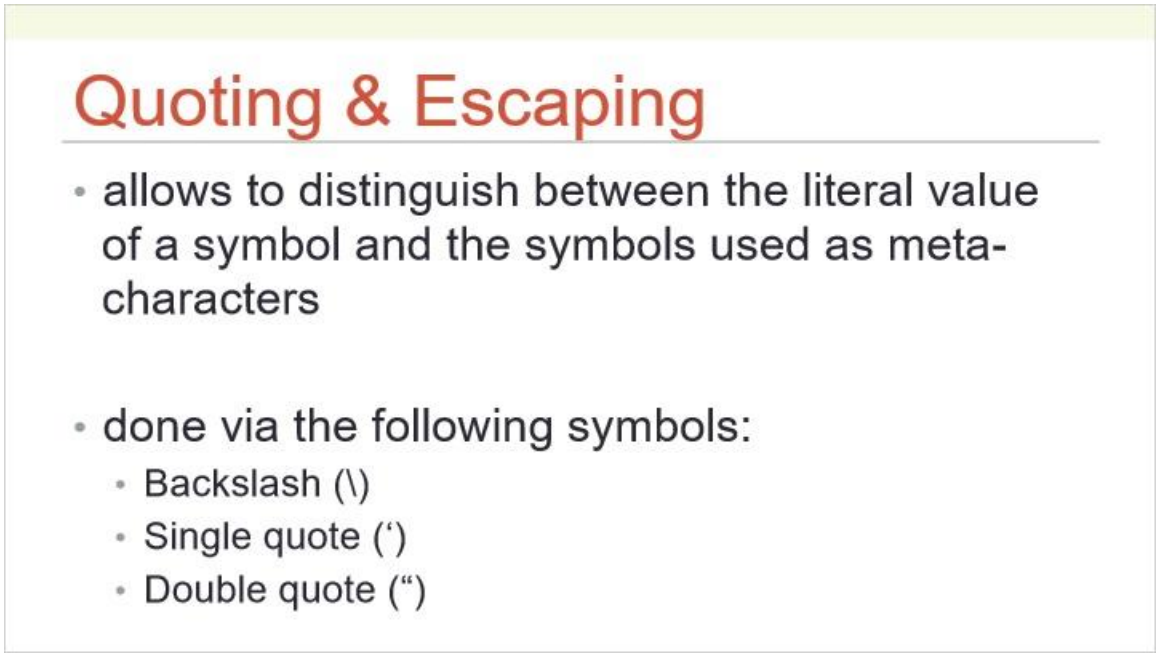
1.20 Regular Expression



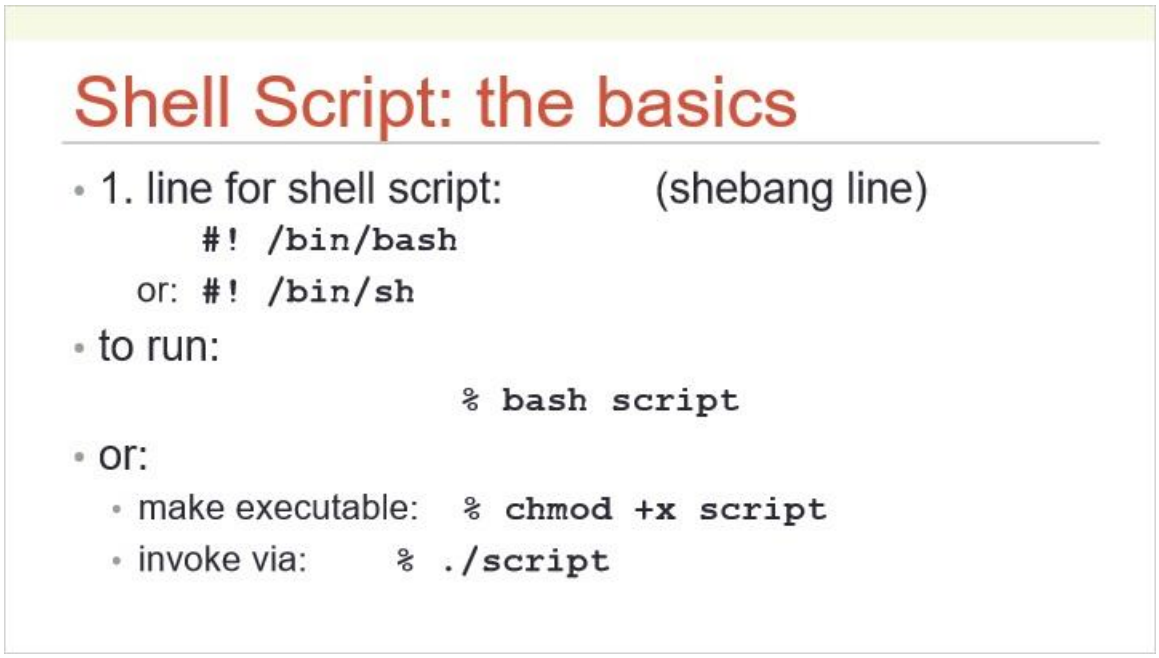
1.21 Metacharacters



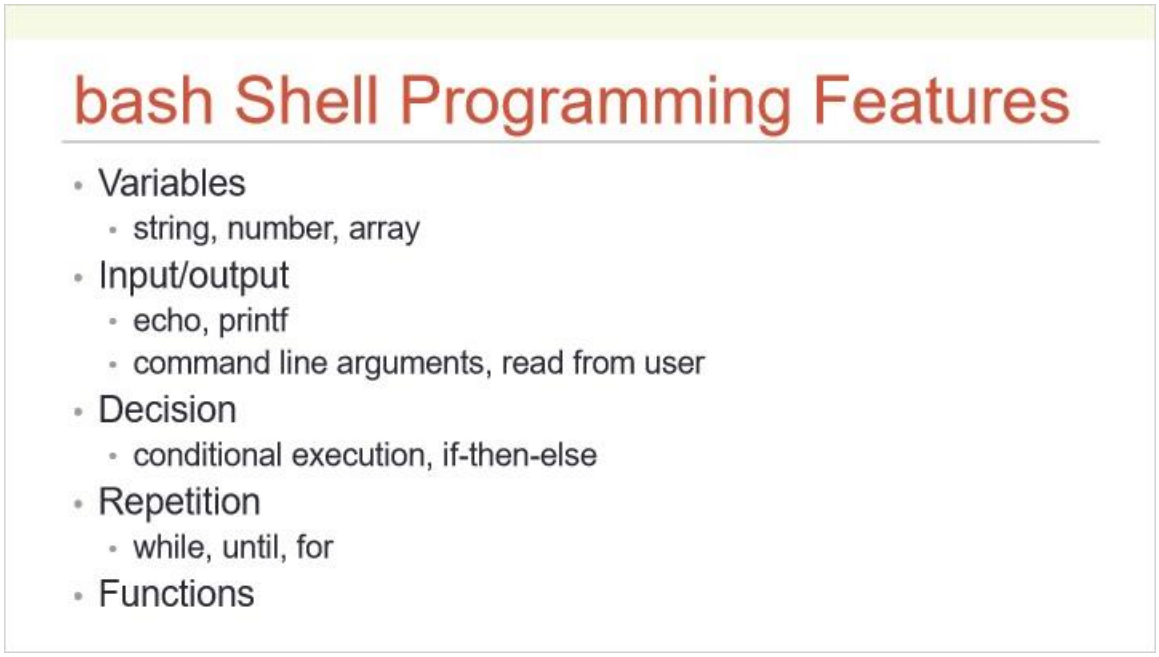
1.22 Quoting & Escaping



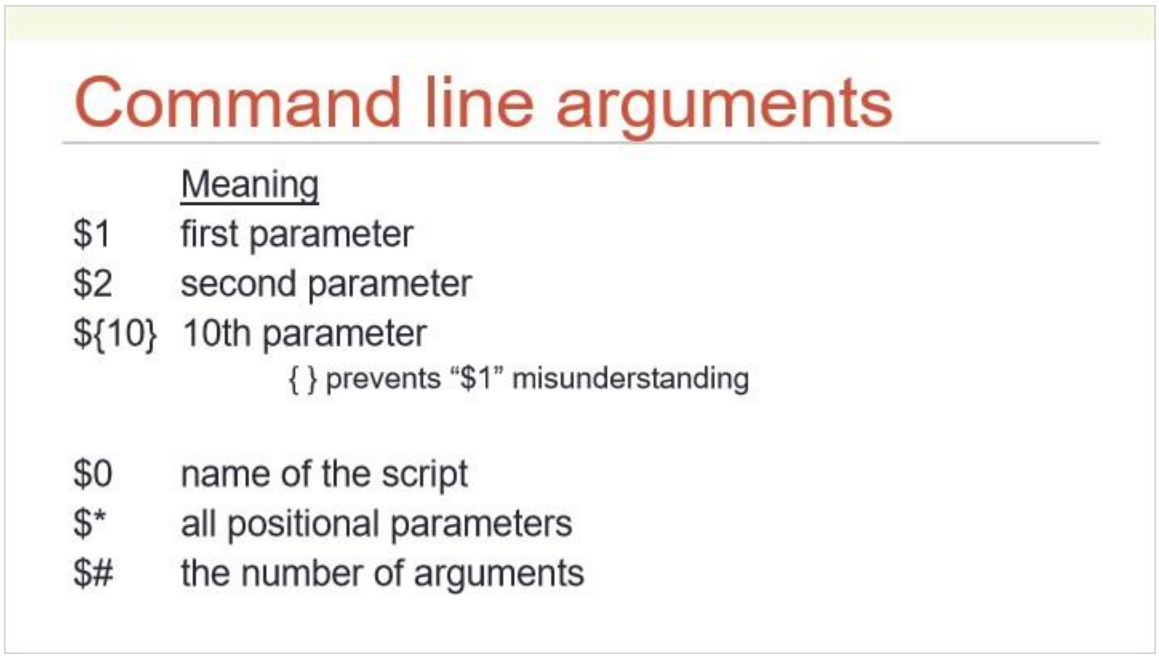
1.23 Shell Script: the basics



1.24 bash Shell Programming Features



1.25 Command line arguments



1.26 User input: read command

User input: read command

Syntax:

```
read [-p "prompt"] varname [more vars]
```

- words entered by user are assigned to **varname** and "**more vars**"
- last variable gets rest of input line

1.27 test command

test command

Syntax:

```
test expression
[ expression ]
```

- evaluates 'expression' and returns true or false

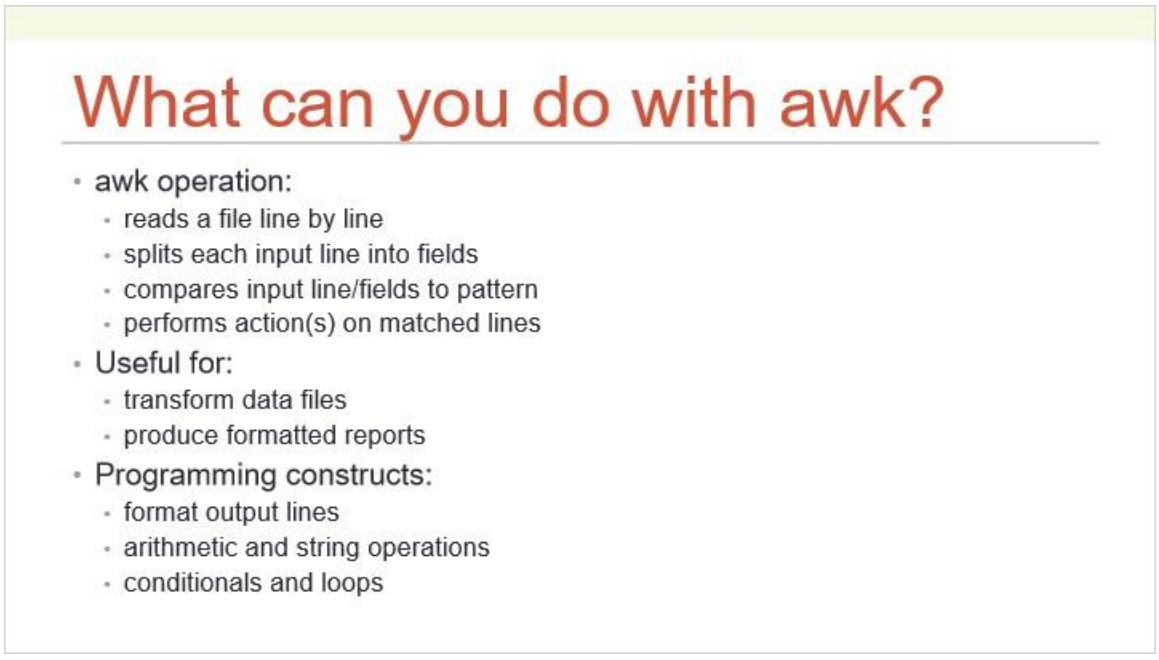
Example:

```
if test $name = "Joe"
then
  echo "Hello Joe"
fi
```

```
if [ $name = "Joe" ]
then
  echo "Hello Joe"
fi
```

1.28 Shell scripting features

1.29 What can you do with awk?



1.30 Basic awk script

Basic awk script

- consists of patterns & actions:

```
pattern {action}
```

  - if pattern is missing, action is applied to all lines
  - if action is missing, the matched line is printed
  - must have either pattern or action

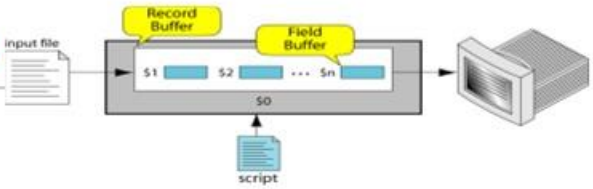
Example:

```
awk '/for/ { print }' testfile
```

- prints all lines containing string "for" in testfile

1.31 awk variables

awk variables



- awk reads input line into buffers: record and fields

- field buffer:
  - one for each field in the current record
  - variable names: $1, $2, …

- record buffer:
  - $0 holds the entire record

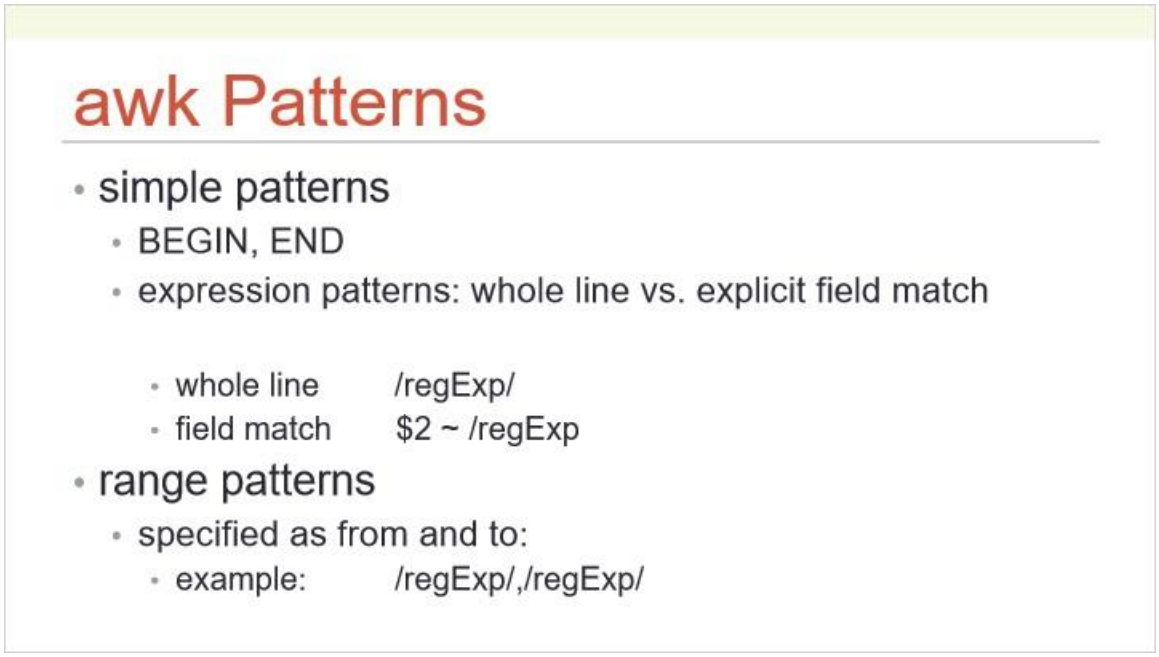
Also:

NR	Number of the current record
NF	Number of fields in current record

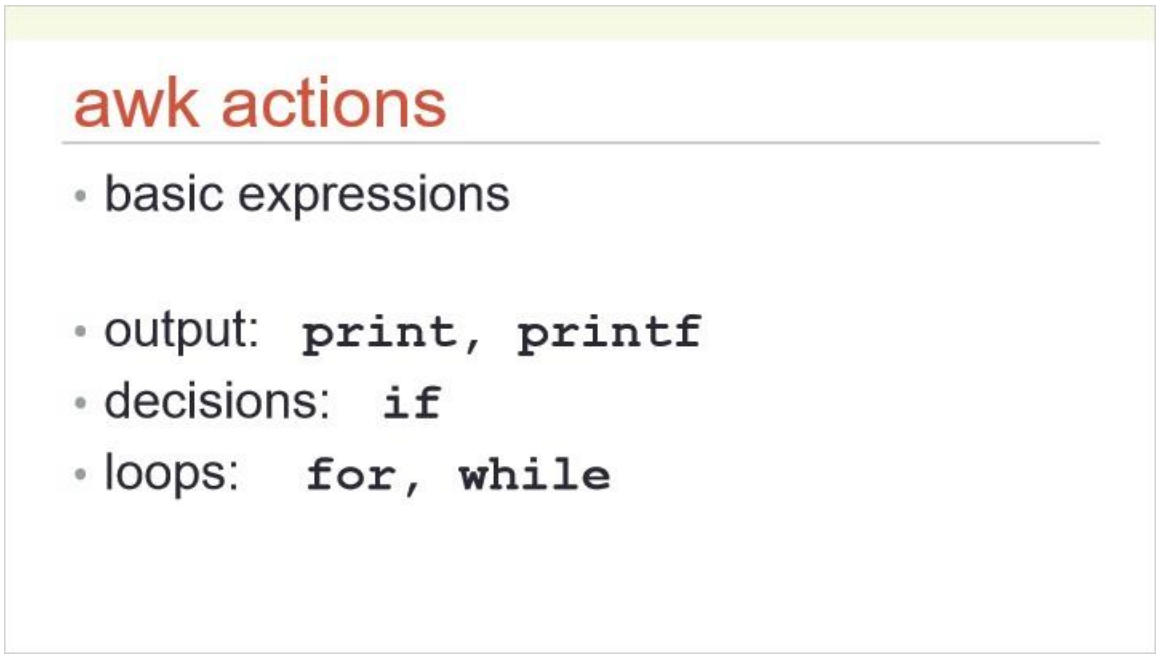
also:

FS	Field separator (default=whitespace)

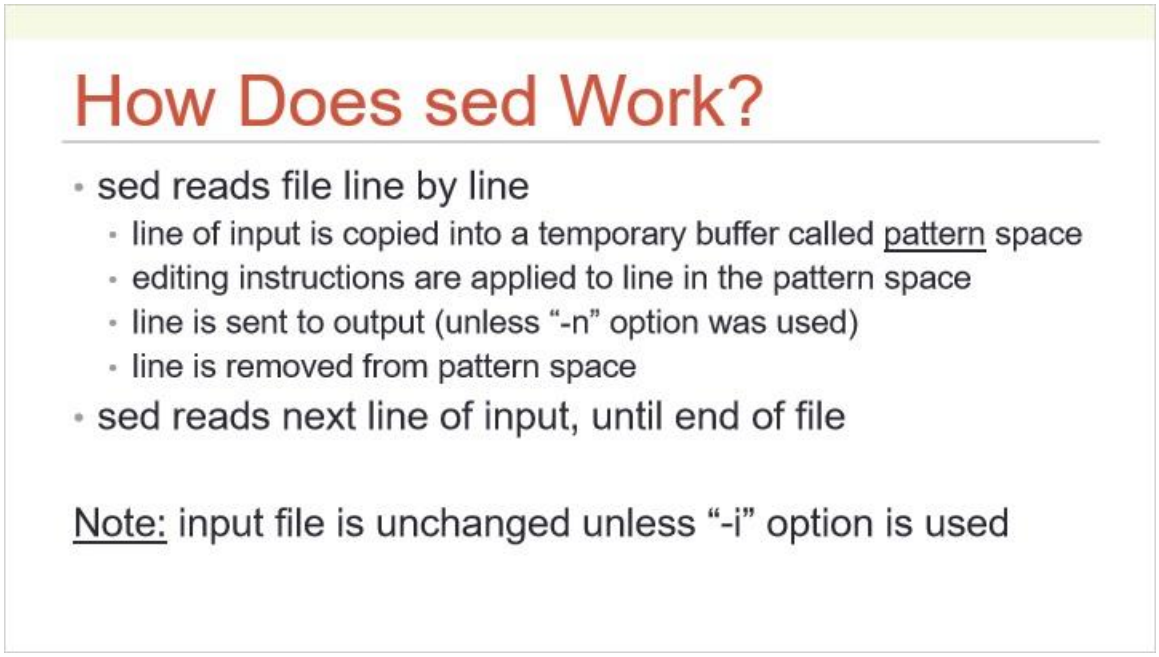
1.32 awk Patterns



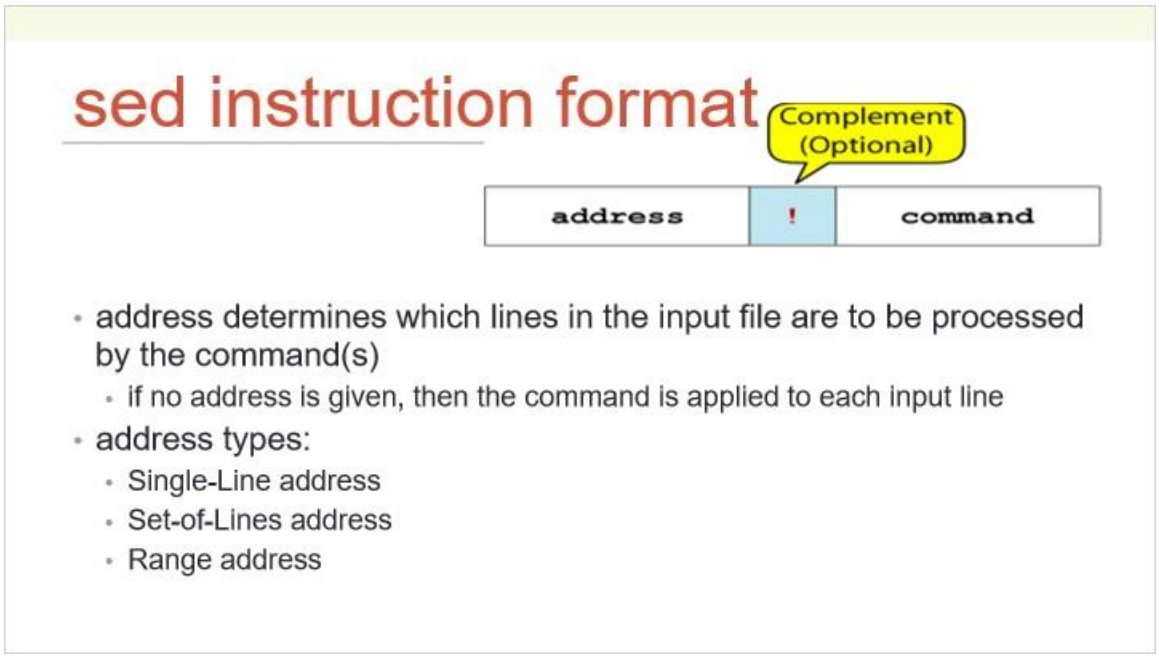
1.33 awk actions



1.34 How Does sed Work?



1.35 sed instruction format



1.36 Topics covered so far