Programming Languages
Linux Commands #3

남 범 석

bnam@skku.edu

grep

- **`grep`** comes from the **`ed`** (Unix text editor) search command "**g**lobal **r**egular **e**xpression **p**rint" or **`g/`***re*`/p`**
- There are two other variants, ***egrep*** and ***fgrep*** that comprise the *grep* family
- *grep* is the answer to the moments where you know you want the file that contains a specific phrase but you can't remember its name
- **`$ grep hh ~/textfile.txt`**

  `beachhead`

  `highhanded`

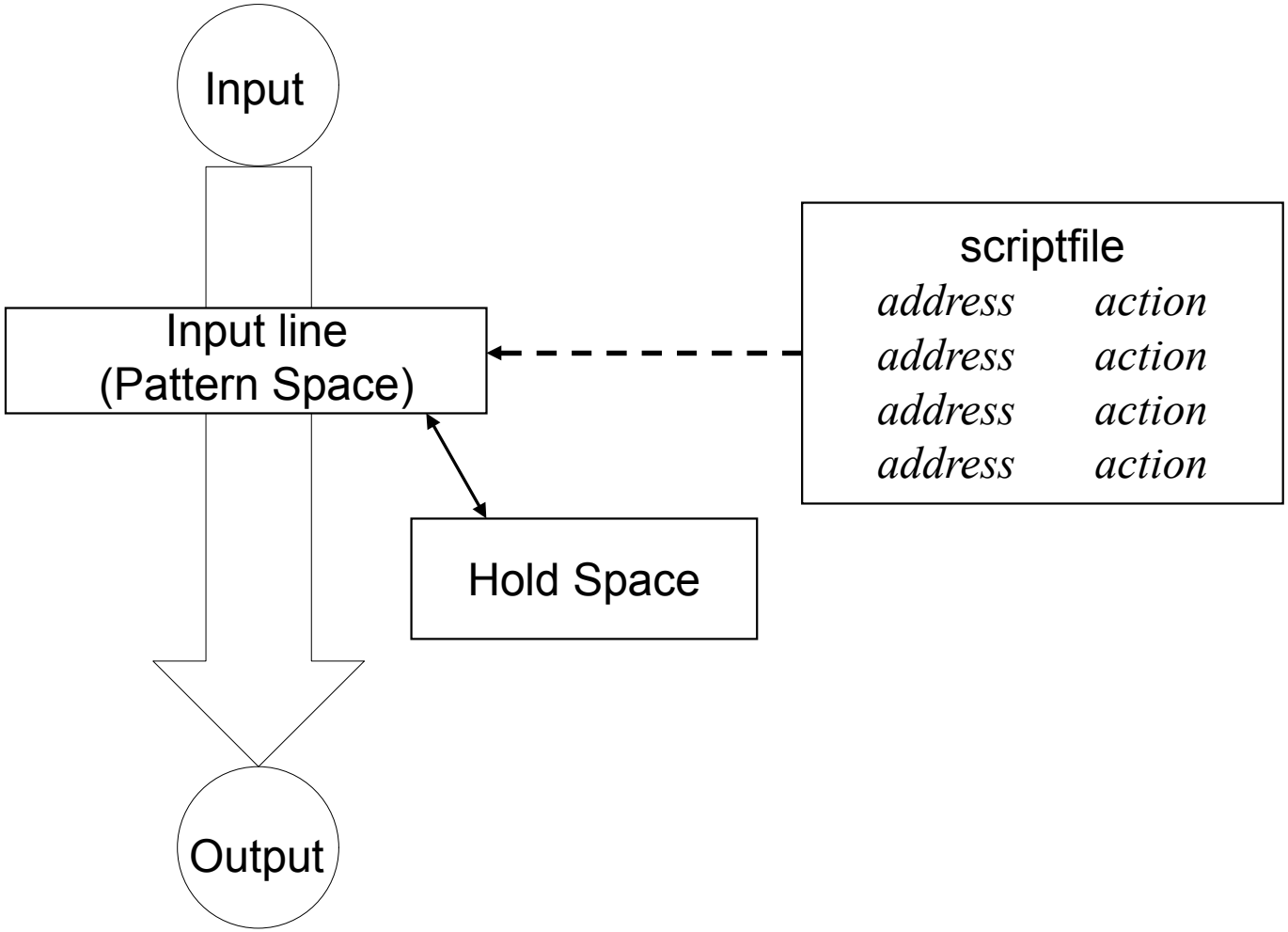
  `withheld`

  `withhold`

sed: Stream-oriented Text Editor

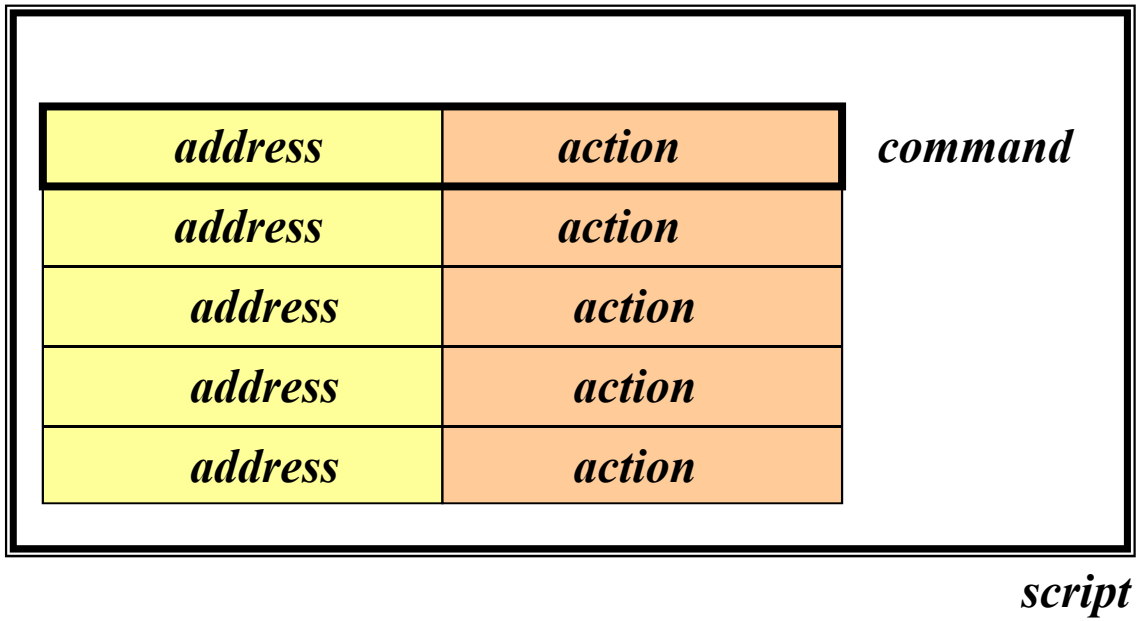
- Look for patterns one line at a time, like **grep**
- *Change* lines of the file
- Non-interactive text editor
  - Editing commands come in as *script*
  - There is an interactive editor *ed* which accepts the same commands
- A Unix filter
  - Superset of previously mentioned tools

sed Architecture



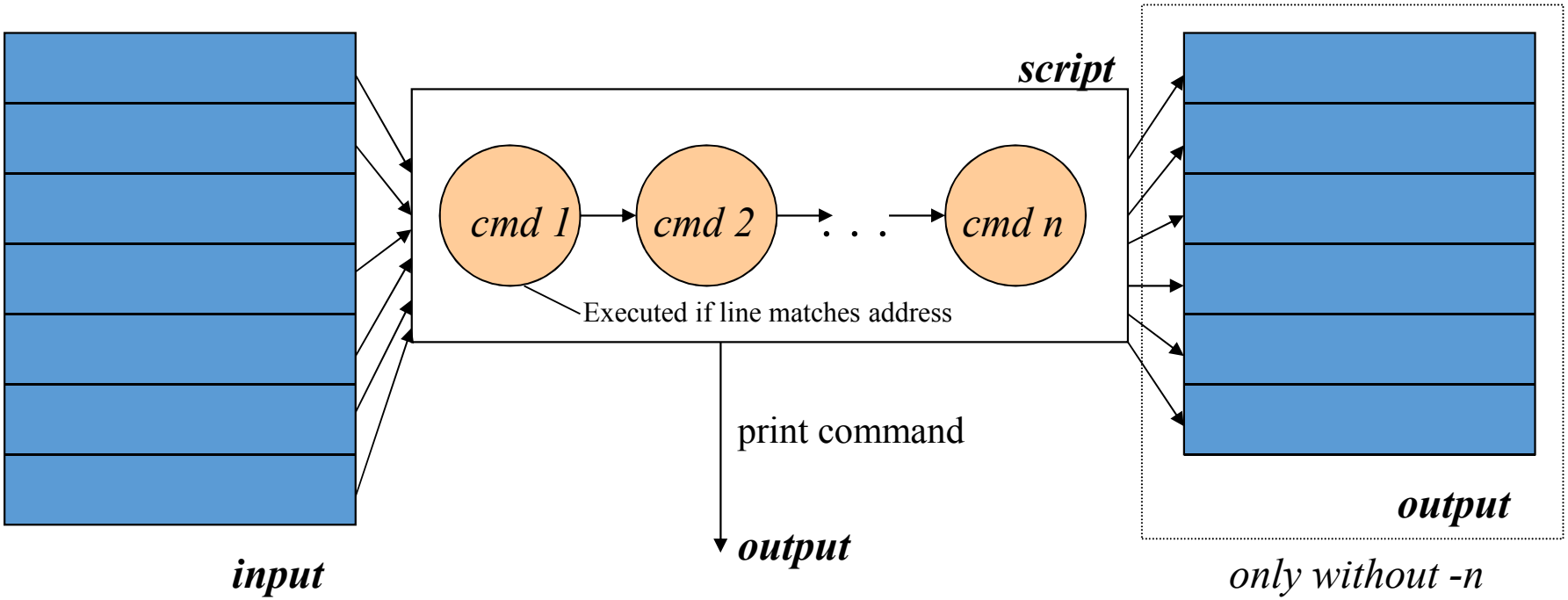
Scripts

- A script is nothing more than a file of commands
- Each command consists of up to two *addresses* and an *action*, where the *address* can be a regular expression or line number.



sed Flow of Control

- *sed* then reads the next line in the input file and restarts from the beginning of the script file
- All commands in the script file are compared to, and potentially act on, all lines in the input file



sed Syntax

- Syntax: *sed [-e] ['command'] [file…]*

  sed [-f scriptfile] [file…]

  - **-e** *command* - the next argument is an editing command rather than a filename

  - ***-f*** *scriptfile* - next argument is a filename containing editing commands

sed Commands

- sed commands have the general form
  - *[address[, address]][!]command [arguments]*
- *sed* copies each input line into a *pattern space*
  - If the address of the command matches the line in the *pattern space*, the command is applied to that line
  - If the command has no address, it is applied to each line as it enters *pattern space*
  - If a command changes the line in *pattern space*, subsequent commands operate on the modified line
- When all commands have been read, the line in *pattern space* is written to standard output and a new line is read into *pattern space*

Addressing

- An address can be either a line number or a pattern, enclosed in slashes (/*pattern*/)
- A pattern is described using *regular expressions* (REs, as in **grep**)
- If no pattern is specified, the command will be applied to **all** lines of the input file
- To refer to the last line: **$**

Addressing (continued)

- Most commands will accept two addresses
  - If only one address is given, the command operates only on that line
  - If two comma separated addresses are given, then the command operates on a range of lines between the first and second address, inclusively
- The ! operator can be used to negate an address, ie; *address!command* causes *command* to be applied to all lines that do ***not*** match *address*

Commands

- *command* is a single letter
- Example: Deletion: **`d`**
- **`[address1][,address2]d`**
  - Delete the addressed line(s) from the pattern space; line(s) not passed to standard output.
  - A new line of input is read and editing resumes with the first command of the script.

Address and Command Examples

- `d` deletes the all lines
- `6d` deletes line 6
- `/^$/d` deletes all blank lines
- `1,10d` deletes lines 1 through 10
- `1,/^$/d` deletes from line 1 through the first blank line
- `/^$/,$d` deletes from the first blank line through the last line of the file
- `/^$/,10d` deletes from the first blank line through line 10
- `/^ya*y/,/[0-9]$/d` deletes from the first line that begins with yay, yaay, yaaay, etc. through the first line that ends with a digit

Substitute

- Substitute: **`s`**
- **`[addr1][,addr2]s/pattern/replacement/flags`**
  - *pattern* - search pattern
  - *replacement* - replacement string for pattern
  - *flags* - optionally any of the following
    - **n** — a number from 1 to 512 indicating which occurrence of *pattern* should be replaced
    - **g** — global, replace all occurrences of *pattern* in pattern space
    - **p** — print contents of pattern space

Substitute Examples

- **`s/Puff Daddy/P. Diddy/`**
  - Substitute P. Diddy for the first occurrence of Puff Daddy in *pattern space*
- **`s/Tom/Dick/2`**
  - Substitutes Dick for the second occurrence of Tom in the *pattern space*
- **`s/wood/plastic/p`**
  - Substitutes plastic for the first occurrence of wood and outputs (prints) *pattern space*

Append, Insert, and Change

- **append** *[address]a*
  text
- **insert** *[address]i*
  text
- **change** *[address(es)]c*
  text
- append/insert for single lines only, not range

- Study by yourself.