

# Command line interface.

Jan Trdlička

Department of Computer Systems  
Faculty of Information Technology  
Czech Technical University in Prague

*trdlicka@fit.cvut.cz*

October 19, 2017

## 1 Discussion of homework

## 2 CLI Parsing Order

- Quoting
- Comments
- Lists, pipelines
- Expansion of special characters
- Word splitting
- Pathname expansion
- I/O redirection
- Command execution

## 3 Homework

# Discussion of homework

- Use only one command `date` to print the current date and the current time on the standard output (terminal) in the following format:

```
Today is Thursday, 05.10.2017 (week 40).  
The time is 14:13:57 [CEST].
```

## Hint

- `man date`
- `export LC_ALL=C`
- Solution
  - In Linux, see `man date`.
  - In Solaris, see `man date` and `man -s 3C strftime`.
  - In MacOS, see `man date` and `man -S 3 strftime`.

```
export LC_ALL=C
```

```
date "+Today is %A, %d.%m.%Y (week %V)%n\  
The time is %H:%M:%S [%Z]."
```

# Discussion of homework - Remote system

- Execute the command from the homework on the server fray1.fit.cvut.cz like user with username USER without permanent connection.

```
ssh USER@fray1.fit.cvut.cz \
'export LC_ALL=C ; \
date "+Today is %A, %d.%m.%Y (week %V)%n\
The time is %H:%M:%S [%Z]."'
```

- Copy the local file /etc/group to home directory on server fray1.fit.cvut.cz.

```
scp /etc/group USER@fray1.fit.cvut.cz:
```

- Verify that the local and transferred files have the same size.

```
ls -l /etc/group
```

```
ssh USER@fray1.fit.cvut.cz 'ls -l $HOME/group'
```

- Can we omit apostrophes in the previous example?

# CLI Parsing Order

- 1 Quoting
- 2 Comments
- 3 Lists, pipelines
- 4 Special characters
- 5 Word splitting
- 6 Pathname expansion
- 7 I/O redirection
- 8 Command execution

# CLI Parsing Order - Quoting

- What is quoting and which characters represent quoting?
  - Quoting is used to remove the special meaning of certain characters or words to the shell.
  - `\, ' ', " "`
- How do you print the following information on the standard output (terminal)? Try to find more correct solutions.

```
Value of variable $HOME: /home/ps1
```

- Solutions

```
echo Value of variable \ $HOME: \ \ \ \ \ "$HOME"
```

```
echo "Value of variable \ $HOME:  $HOME"
```

```
echo Value of variable ' $HOME:  ' "$HOME"
```

```
printf "Value of variable \ $HOME  %s\n" "$HOME"
```

# CLI Parsing Order - Quoting

- How do you print the following information on the standard output?  
Try to find more correct solutions.

```
Output of command pwd: /etc
Value of variable $PWD: /etc
```

- Solutions

```
echo -e \
"Output of command pwd:      $(pwd)\n\
Value of variable \ $PWD:    $PWD"
```

- Use "type echo" to get info what "echo" is.
- Use "man bash" (not "man echo") or "help echo" to get help for command echo.
- Use command printf instead of command echo.

```
printf \
"Output of command pwd:      %s\n\
Value of variable \ $PWD:    %s\n" \
"$(pwd)" "$PWD"
```

# CLI Parsing Order - Comments

- What character represents a comment in the shell?
  - The character # indicates that the remainder of the line is a comment when found as the first character of a word.

```
echo "$HOME"      # echo prints path to your home
```

```
echo "# it isn't comment" # it is comment
```

```
echo where is# my comment \?
```



# CLI Parsing Order - Lists, pipelines

- Which characters can separate commands in the shell and what is their meaning?

- ; sequential execution,
- & parallel execution,
- | pipeline.

- Run the following commands sequentially (one by one):

`sleep 10, date, hostname.`

```
date ; sleep 10 ; hostname
```

- Run the following commands in parallel: `sleep 10, date, hostname.`

```
date & sleep 10 & hostname &
```

- Run a web browser in background.

```
google-chrome &
```

- Print the output of command `"ls -lR /"` by command `less`.

```
ls -lR / | less
```

# CLI Parsing Order - Expansion of special characters

- What special characters do you know?

- `~`, `' '`, `$`, `$ ( )`, `$ ( ( ) )`, `{ }`.

- Tilde expansion `~`

- Print the path to your home directory on standard output.

```
echo ~
```

- Print the path to the home directory of the user `muzikar` on standard output.

```
echo ~muzikar
```

# CLI Parsing Order - Expansion of special characters

- Command substitution ``` or `$( )` (newer syntax)
  - Save the kernel name and the kernel release to the shell variable `OS`. Hint: use command `uname`.

```
OS="`uname -sr`"
```

```
OS="$(uname -sr)"    # Better syntax
```

- Can we omit the double quotes in the previous example?
- Save the number of processes running on the system to the shell variable `LOAD`. Hint: use pipe of command `ps`, `tail`, and `wc`.

```
LOAD="$(ps -e | tail -n +2 | wc -l)"    # Linux
```

```
LOAD=$(ps -e | tail +2 | wc -l)        # Solaris
```

- Can we omit the double quotes in the previous example?

# CLI Parsing Order - Expansion of special characters

- Parameter (Variable) expansion \$

- Print the contents of the variables OS and LOAD from the previous slide on the standard output by one command and one variable per one line.

```
echo -e "OS=$OS\nLOAD=$LOAD"
```

```
printf "OS=%s\nLOAD=%s\n" "$OS" "$LOAD"
```

- Can we omit the double quotes in the previous example?
- Use the command `mkdir` to create a subdirectory `bin` in your home directory.

```
mkdir "$HOME/bin"
```

- Add path to this directory to the shell variable `PATH`.

```
export PATH="$PATH:$HOME/bin"
```

- Can we omit the double quotes in the previous example?

# CLI Parsing Order - Expansion of special characters

- Arithmetic expansion `$(( ))`

- Create a command that 5 hours after execution writes a message "Hello word" on standard output. Hint: use command `sleep` and arithmetic expansion for timeout.

```
sleep $((5*60*60)) ; echo "Hello word"
```

- Brace expansion `{ }`

- Brace expansion is a mechanism by which arbitrary strings may be generated.
- Create (set modify time to) the following files by command `touch`:  
f1, f1.txt, f1.c, f2, f2.txt, f2.c, f3, f3.txt, f3.c

```
touch f{1,2,3}{,.txt,.c}
```

- Print all integer numbers from 1 to 100 on the standard output.
  - See "man bash" and search for a string "brace expansion".

```
echo {1..100}
```

# CLI Parsing Order - Word splitting

- The shell treats each character of shell variable IFS (Internal Field Separator) as a delimiter, and splits the results of the other expansions into words on these characters.
- The default value of IFS is space, tab, newline.
- Try the following commands and explain their different behavior.

```
echo $(ls -l)
```

```
echo "$ (ls -l) "
```

# CLI Parsing Order - Pathname expansion

- Which characters represent pathname expansion and what is their meaning?
  - \*, ?, [ ], [! ], [^ ]
- Explain the behavior of the following commands?

```
cd /usr/bin  
ls *q
```

```
ls q*
```

```
ls *q*
```

```
ls */*
```

```
ls ?
```

```
ls ??
```

# CLI Parsing Order - Pathname expansion

- Explain the behavior of the following commands?

```
ls [a-j]?
```

```
ls [!a-j]?
```

```
ls ?[^a-j]
```



# CLI Parsing Order - I/O redirection

- Which characters represent I/O redirection and what is their meaning?
  - <, >, >>, <<END
- Explain the following commands?

```
wc -l /etc/group
```

```
wc -l < /etc/group
```

```
ls -l > f1 ; cat f1
```

```
date > f1 ; cat f1
```

```
date >> f1 ; cat f1
```

```
cat <<END >f1
```

```
Current date: $(date) in directory $PWD.
```

```
Login name: $USER
```

```
END
```

# CLI Parsing Order - Command execution

- If the command name contains no slashes (only name, no path).  
How can the shell locate it?
  - If the name is neither a shell function nor a builtin, and contains no slashes, the shell searches each element of the PATH for a directory containing an executable file by that name.
  - The key TAB works like command completion in bash.
- The following commands can be used to see how each command name would be interpreted by the shell.

```
type ls
type -p ls
/usr/ucb/whereis ls
/usr/bin/which ls
echo $PATH
```

- Try the following commands and explain their behaviour.

```
echo PWD is $PWD
```

```
echo PWD\ is\ \ $PWD
```

```
echo "PWD is      $PWD"
```

```
echo "\$PWD is      $PWD"
```

```
echo '\$PWD is      $PWD'
```

# Homework

- Try the following commands and explain their behaviour.

```
cmd=who
```

```
echo cmd
```

```
echo $cmd
```

```
echo "$cmd"
```

```
echo '$cmd'
```

```
echo '$cmd'      # or echo "$($cmd)"
```

```
$cmd
```

- How can we protect output format of the command `who`?