GlobalLogic

A Hitachi Group Company

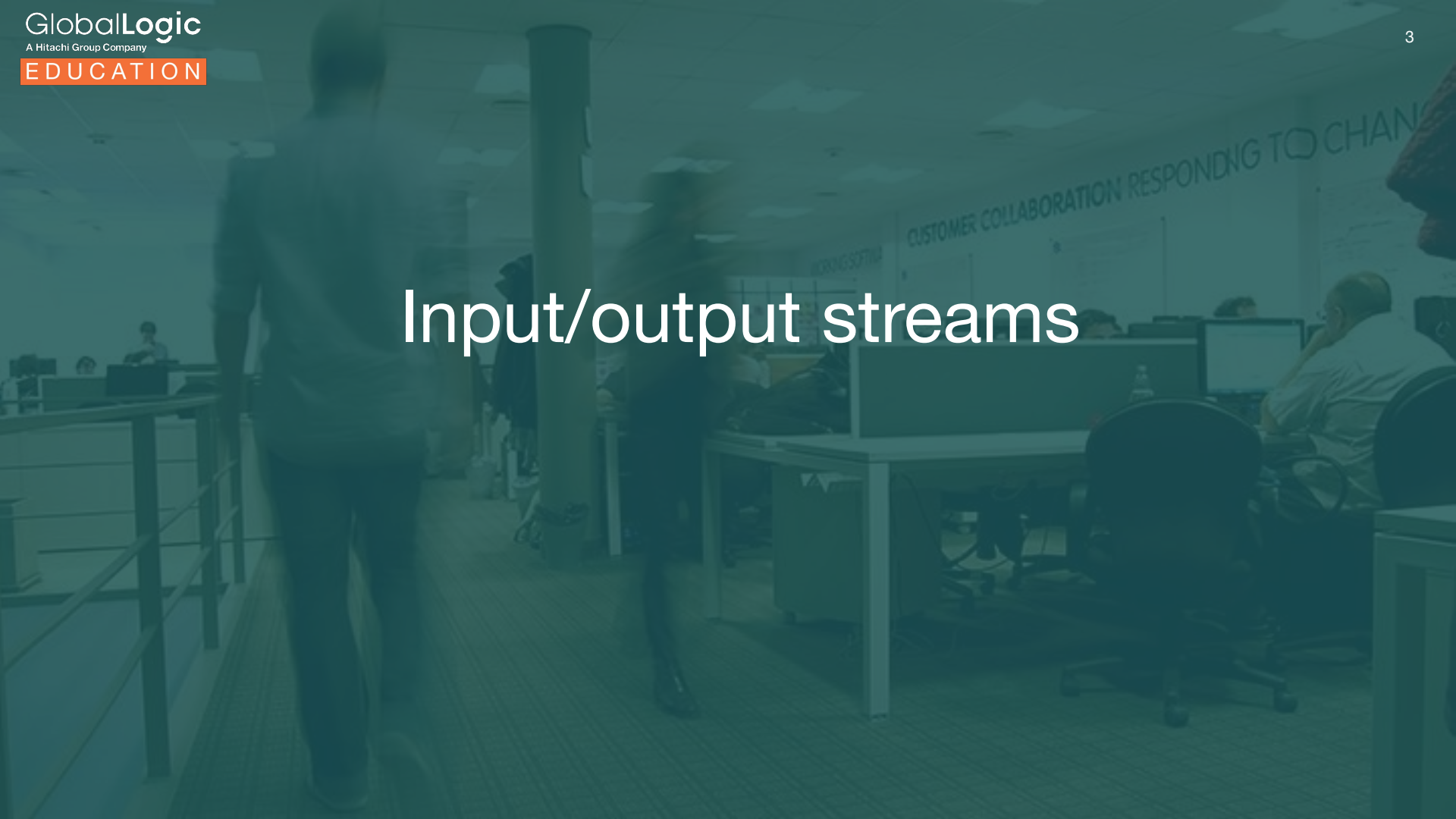
EDUCATION

Smart Start: Linux/Networking
Linux command line intro. Part 2

Sergii Kudriavtsev

Agenda

1. Input/output streams
2. Pipes
3. wildcards
4. vim
5. bash / sh
6. bash variables
7. special variables
8. Eval, $(), ``, xargs



Input/output streams

- Input / Output Streams
  - stdin (0)
    - <
      - `$ read VARIABLE < file.txt`

- Input / Output Streams
  - stdout (1)
    - >
      - `$ cat file1.txt | grep regexp > file2.txt`
      - `$ cat file1.txt > file2.txt | grep regexp`
      - `$ cat file1.txt >/dev/null 2>&1 | grep regexp > file2.txt`
      - `$ cat file1.txt 2>&1 >/dev/null | grep regexp > file2.txt`
      - `$ command > /dev/null`
      - `$ cat > a << EOF`
      - `$ command < input-file > output-file`
      - `$ : > filename`
      - `$ > filename`
      - `$ {echo 1; echo 2; echo 3;} > file`
    - >>
      - `$ cat file1.txt >> file2.txt`
    - << MARKER - end of input
      - ```
        $ cat > file1.txt << end_of_file
        line1
        line2
        line3
        end_of_file
        ```
        ```

- Input / Output Streams
  - stderr (2)
    - 2>
      - `$ command 2> /dev/null`
      - `$ command > file_stdout.txt 2> file_stderr.txt`
    - 2>&1
      - `$ command > /dev/null 2>&1`
  - `$ &>filename #stdout + stderr`
  - `$ i>&j # File with handle i to j is redirected.`
    `# Output to the file with the descriptor i is transferred`
    `# to the file with the descriptor j.`
  - `$ >&j # The file with handle 1 (stdout) is redirected to file with handle j.`
  - `$ exec 3 <> File # Open "File" and associate with handle 3.`
  - `$ exec 3>&- # Close handle 3.`
  - /dev/tty
    - This is a special file, representing the terminal for the current process

Pipes

- Pipes
  - Explanation
    - one-directional
    - connects **stdout** of the left process to the **stdin** of the right process
    - does not connect **stderr** of the left process
  - Examples
    - `$ cat file.txt | grep value`
    - `$ cat file.txt | grep value1 | grep value2`
    - `$ echo $PATH | tr ':' '\n' | grep /usr`
    - `$ cat file.txt | sort`

Wildcards

- Shell wildcards
  - * - matches string of any characters or empty string. Examples:
    - pineapple
    - e373
  - ? - matches single character. Examples:
    - a
    - Z
    - x

- Shell wildcards
  - [...] - matches any one of the enclosed characters
    - [a]. All of the possible values:
      - a
    - [abc]. All of the possible values:
      - a
      - b
      - c
    - [a-c]. All of the possible values:
      - a
      - b
      - c
    - [a-c4X-Z]. All of the possible values:
      - a
      - b
      - c
      - 4
      - X
      - Y
      - Z

- Pathname expansion: wildcards, globs
  - https://en.wikipedia.org/wiki/Glob_(programming)
    - https://en.wikipedia.org/wiki/Glob_(programming)#Unix
  - * - matches any string, including null string
    - `$ ls *.txt # file1.txt, file2.txt, file3.txt will match`
  - ? - matches any single character
    - `$ ls file?.txt # file1.txt, file2.txt, file3.txt will match`
  - [*abcde*], [*a-e*], [*abcdek-m*], [*a-ek-m*] - matches any one of the enclosed characters
    - `$ ls file[123].txt # file1.txt, file2.txt, file3.txt will match`
    - `$ ls file[1-3].txt # file1.txt, file2.txt, file3.txt will match`
    - `$ ls file[13].txt # file1.txt, file3.txt will match`
    - `$ ls file[A-Za-z0-9].txt # fileX, filex, file1.txt, file2.txt, file3.txt will match`
    - `$ ls test_sbit.[A-Z]`
  - [!*abcde*], [!*a-e*], [!*abcdek-m*], [!*a-ek-m*] - matches any one of the NOT ENCLOSED characters

- Pathname expansion: wildcards, globs
  - `shopt`
    - `$ shopt -s option_name`
    - `$ shopt -u option_name`
    - `nocaseglob -` controls case sensitivity for path expansion
    - `dotglob -` controls path expansion for hidden files
  - For pathname expansions better use C locale to get predictable results:
    - Check current locale
      - `$ locale`
    - Set collate
      - `$ unset LC_ALL; export LC_COLLATE=C`
      - `$ unset LC_ALL; export LC_COLLATE=C.UTF-8`

vi/vim

- editing
  - `EDITOR` env variable
    - `$ export EDITOR=mcedit`
  - `$ vi file.txt`
    - editing
      - I, i, A, a
      - x, dw, db, dd
      - Esc
    - searching
      - /pattern
      - ?pattern
    - saving
      - :w
    - exiting
      - :q
      - :q!
      - :x
      - :wq
    - key bindings
      - http://www.viemu.com/vi-vim-cheat-sheet.gif
    - Vim: vimtutor (vim must be installed also)

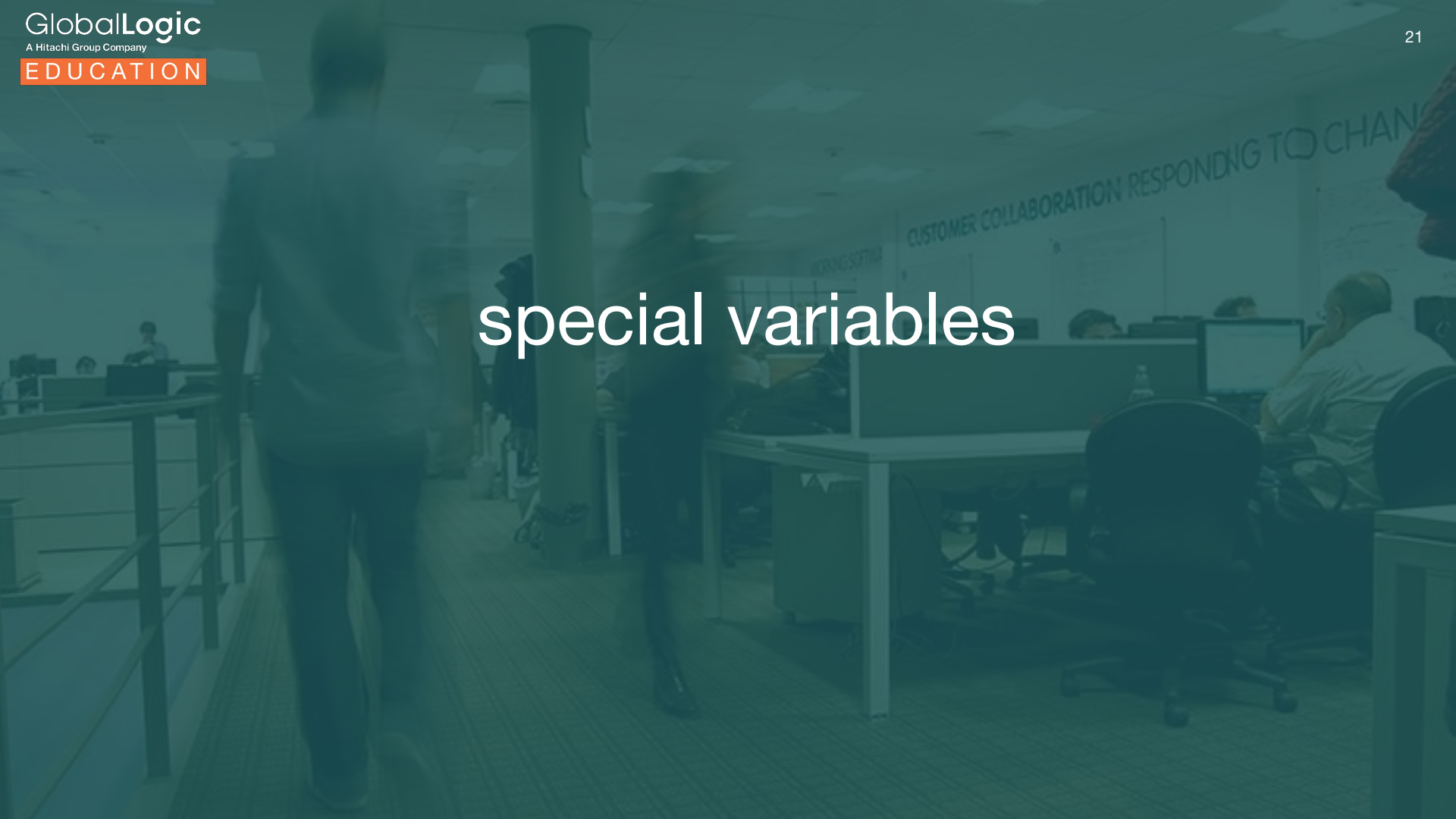
bash/sh

- Bash options
  - `$ set -o`
    - `$ set +o allexport # disable exporting variables (Also +a option)`
    - `$ set -o allexport # enable exporting variables (Also -a option)`
  - `$ shopt # Bash only`
    - `$ shopt -s nocaseglob # SET:   case-insensitive path expansion`
    - `$ shopt -u nocaseglob # UNSET: case-insensitive path expansion`
- Bash history
  - `$ SAVEHIST=100000`
  - `$ HISTFILE=$HOME/.:qhistory`
  - `$ HISTSIZE=100000`
  - arrow keys (ctrl-p/ctrl-n)
  - ctrl-R - find previous command, ctrl-S - find next command (`stty -ixon`)
  - $HOME/.history
  - $HOME/.bash_history
  - `$ history`
    - `$ !command_num`
    - `$ !! # run previous command`
    - `$ sudo !! # run previous command as root`
  - `$ history -c # Clear history`
  - `$ history n #Display n lines of history`

- Configuration files
  - Run upon login
    - **/etc/profile**
    - $HOME/.login, **$HOME/.profile**
  - Run upon interactive shell creation
    - **/etc/bash.bashrc**
    - **$HOME/.bashrc**, $HOME/.kshrc
  - sourcing (. command)
    - `$ . ./.profile #bash and ksh`
    - `$ source ./.profile #bash`
    - `$ source $HOME/bin/my_script.bash`
  - executing (DOES NOT AFFECT CURRENT SHELL PROCESS)
    - `$ chmod +x $HOME/bin/my_script.bash`
    - `$ $HOME/bin/my_script.bash`
    - mount options of file system where the script is located
      - `exec`
      - `noexec`

bash variables

- Shell Variables
  - Names
    - Characters: A-Z, a-z, 0-9, _
    - case sensitive
    - Better use UPPER CASE
  - Values
    - Set
      - `$ VARIABLE=value`
    - Unset
      - `$ unset VARIABLE`
    - View
      - `$ echo $VARIABLE`
      - `$ echo ${VARIABLE}`
      - `$ env # show exported variables (environment variables)`
      - `$ set # show all variables and functions (bash and ksh)`
        - `$ declare # show all variables and functions (bash)`



special variables

- Shell Variables
  - Special variables
    - HOME env variable
    - PATH env variable
    - SHELL env variable
    - PS1, PS2, PS3, PS4 - prompt
    - ? - exit code from the last command. 0 - successful, otherwise - unsuccessful
    - $ - PID of the current shell process

- Shell Variables
  - Types and scope
    - exported / not exported
      - `$ export VARIABLE`
      - `$ export VARIABLE=value`
    - arrays
      - indexed
        - `declare -a ARRAY` # bash
        - `typeset -a ARRAY` # ksh, bash
          - `$ ARRAY[0]=value0`
          - `$ ARRAY[1]=value1`
          - `$ echo ${ARRAY[0]}`
            `value0`
          - `$ echo ${ARRAY[*]}`
            `value0 value1`
          - `$ echo ${#ARRAY[*]}`
            `2`
          - `$ echo ${!ARRAY[*]}`
            `0 1`

- Shell Variables
  - Types and scope
    - associative
      - `declare -A ARRAY_NEW` # bash
      - `typeset -A ARRAY_NEW` # ksh, bash
        - `$ ARRAY_NEW[key7]=value0`
        - `$ ARRAY_NEW[ppp]=another_value`
        - `$ echo ${ARRAY_NEW[ppp]}`
          `another_value`
        - `$ echo ${ARRAY_NEW[*]}`
          `value0 another_value`
        - `$ echo ${!ARRAY_NEW[*]}`
          `key7 ppp`
    - integers
      - `declare -i INTEGER` # bash
      - `typeset -i INTEGER` # ksh, bash
      - `$ INTEGER=$(( 2 + 3 ))`
      - `$ INTEGER=$(( 2 + $INTEGER ))`
      - `$ INTEGER=aaa # INTEGER=0`
      - `[[ "$INTEGER" -eq 0 ]] && echo equals`
      - `$ man bash`
        - Search for "Arithmetic Expansion"

- Shell Variables
  - Types and scope
    - double
      - echo "scale=9; 33/100.1" | bc
    - Links

```
a=letter_of_alphabet
letter_of_alphabet=z
echo "a = $a"
echo "Now a = ${!a}"
eval c=\$$a
echo $c
```

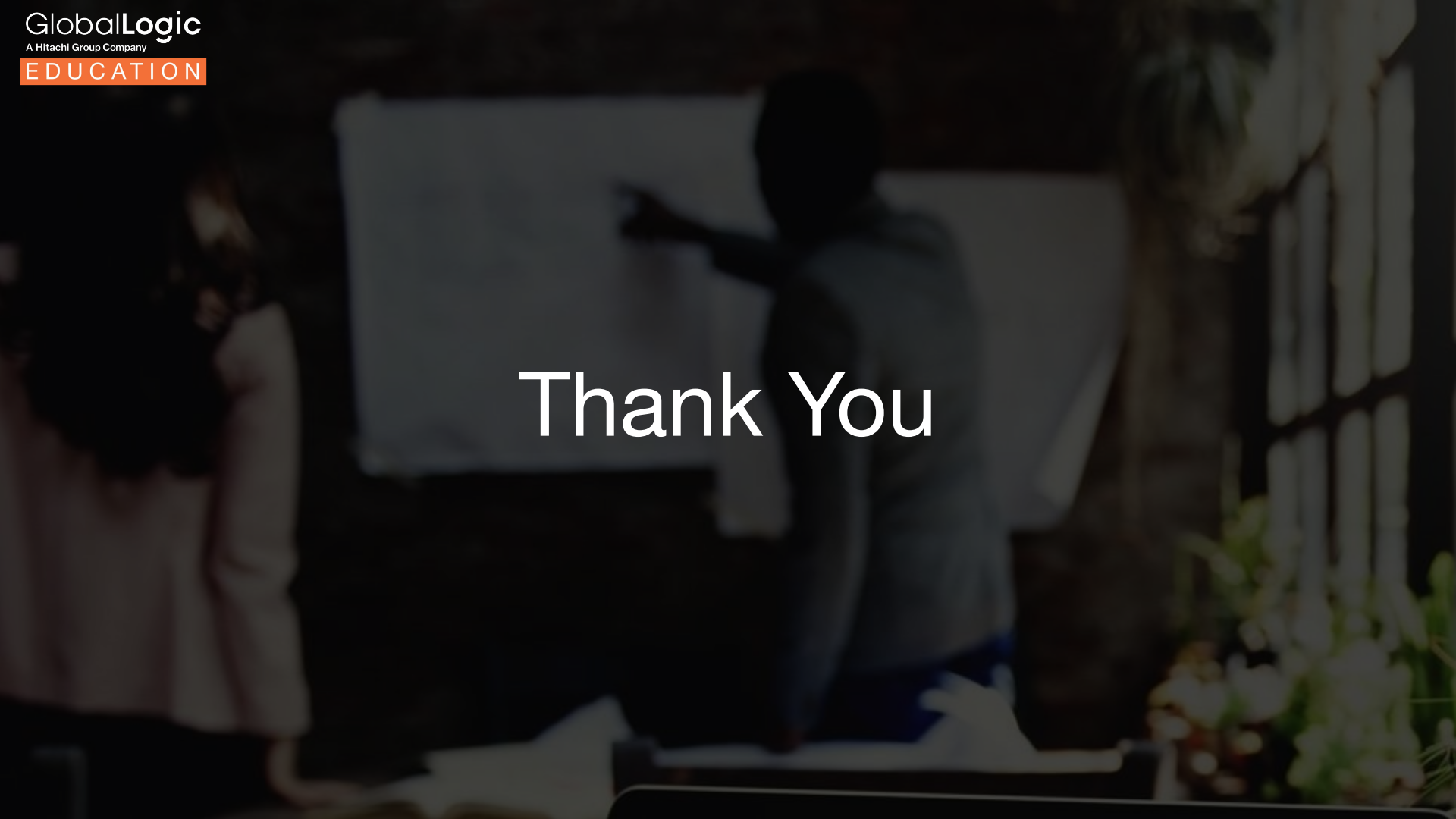
- Shell Variables
  - Types and scope
    - Data type magic

```
a=2334
let "a += 1"
echo "a = $a "          # a = 2335
echo $(( a * b ))       # 0
b=
let "a = b + a" #a = 2335
echo b${b}b             #bb
let "b = b * 2"
echo $b                 # 0
b=blabla
d=dada
echo "$(( b + (d + a) ))" #2335
[[ 16 > 0x10 ]] && echo OK #OK
[[ 16 -eq $((0x10)) ]] && echo OK #OK
```

Eval, $(), ``, xargs

- constructing with eval
  - `$ eval "command arg1 arg2"`
    - `$ eval "echo export VAR1=value1"`
    - `$ eval "`ssh-agent -s`"`
    - `$ a=”la | more”`
    - `$ echo $a`
    - `$ $a`
    - `$ eval $a`
    - `a=b b=c ; eval echo \$$a ### shall produce "c"`

- constructing with xargs
  - `$ find ~ -name '*.txt' | xargs tar cvf archive_text.tar`
    - `$ tar cvf archive_text.tar `find ~ -name '*.txt'``
  - `$ find ~ -name '*.tmp' | xargs rm -f`
    - `$ rm -f `find ~ -name '*.tmp'``
  - `$ ls -1 | xargs ls -l`
    - `$ ls -l `ls -1``


GlobalLogic
A Hitachi Group Company
EDUCATION
Thank You