# Session 2 - Style and more.

## Topics

- Review session 1
- Enable file suffix
- Loops
- Dos & Donts

## What happened previously. . . ?

Go through the most important parts of the last session: * Variables * echo * IF statements * GOTO

Also discuss the homework assignment.

## Intermission:

### Programming Dos & Donts

## Programming Dos & Donts

0. English.
1. Pretty please.
2. Echo loud and clear.
3. Don't trust user input ;)
4. Document it!

## 0. English:

Do all technical naming in English:

```
* variables
* functions
* comments
* ...
```

Text output may of course be in any language.

## 0. English: Why?

- Avoids illegal characters
- Avoids language/parsing errors
- Facilitates code exchange and understanding ("RVB" anyone?)

"RVB" as variable for pixel color instead of "RGB". Why? **R**ouge, **V**ert, **B**leu French coder. It made it harder for me to debug/improve his code :(

## 1. Pretty please:

- Proper naming (vars, etc)
- Indent!
- Linebreaks.
- Quotes for strings.
- Be consistent.

## 2. Echo loud and clear:

Loglevel?

- silent
- verbose
- debug

Short and superficial introduction to how to echo what's happening and what "loglevel" or "verbosity" means.

## 3. Don't trust user input ;)

- Check it.
- Escape it.
- Or forge(t) it.

## Check input:

- Does file exist?
- Is this a folder?
- Does it contain illegal characters?
- Is it empty?
- etc.

## Escape input:

- Disarm special characters.
- "Escape character"
- Often backslash "\"

## Forge / forget it:

- Masks and placeholders
- Concatenate values/strings
- If input doesn't make sense:
  Forget it. Bail bout.
  But let the user know "*why*".

Example: Create a string/parameter by concatenating user input in a way that reduces security/error options.

Such as: User input: "ffv1" instead of "-c:v ffv1"

Has pros/cons of course, but we're still learning the first steps here :)

## 4. Document it!

- You think you'll remember?
- Clear to you = clear to others?

## Clear?

**Shell script:**

```
i=1
for f in *.dpx; do
    mv $f $(printf %07d $i).dpx
    i=$((i+1))
done
```

## Clear.

**Shell script:**

```
# Set start number of output files:
i=1

# Iterate through all DPX image files:
for FILE in *.dpx; do
    # zero-pad index to 7 digits:
    INDEX=$(printf "%07d" $i)
    mv $FILE $INDEX.dpx
    # Increment counter:
    i=$((i+1))
done
```

## Oh. One more thing:

### Windows & the hidden file extension.

## The hidden file suffix

- "hello.bat" is displayed as "hello"
- Windows Explorer hides the suffix by default.
- This is bad. Very bad.
- It's better to see what you're dealing with.

To enable it: * Open Windows Explorer. * Press and release "Alt" key to show menu. * Menu: *Tools* > *Folder Options* > *View* * Uncheck: "Hide extension for known file types" * Press "OK"

## Back to coding!

## Environment Variables

Some *variables* are already configured in your *environment*. For your convenience.

Examples: Where's the program folder? Where's my home?

## Environment Variables

**Batchfile:**

```
set
echo %ProgramFiles%
```

**Shell script:**

```
printenv
env
echo "$HOME"
```

env vs printenv? See question #123473 on stackexchange.com

## Looooooops!

> "Do something, *until / while / for* something. Then move on."

## FOR loop only. Here.

Here's the basic idea of a *FOR loop*:

```
for X in Y do ...
```

Sufficient for most things :)

## FOR: Counter

```
:: IN (Start, Stepsize, Maximum)
FOR /L %%I IN (0, 2, 8) DO echo Index is: %%I
echo.
```

## FOR: Static set

```
:: IN (a b c d e ...)
FOR %%I IN (1 2 3) DO echo Index is: %%I
echo.
```

## FOR: Command arguments

```
:: %* = %1 %2 %3 ...
FOR %%F IN (%*) DO echo Params: %%F
```

## FOR: Files in folder

**Batch file:**

```
@echo off
FOR /R "C:\Videos\" %%F IN (*.avi) DO (
    echo %%~nF
)
pause
```

**Shell script:**

```
for FILE in ~/Videos/*.avi; do
    echo "${FILE%.*}"
done
```

The “echo” statement here outputs the *basename* of all files matching the filemask (*.avi) - without their suffix.

## Loop references

### Batch:

- Tutorial 1 (Steven Jansen)
- Tutorial 2 (TryToProgram.com)
- FOR examples (ss64.com)

## String modifiers (DOS/WIN)

```
Variable with modifier  Description

%~I                     Expands %I which removes any surrounding
                        quotation marks ("").
%~fI                    Expands %I to a fully qualified path name.
%~dI                    Expands %I to a drive letter only.
%~pI                    Expands %I to a path only.
%~nI                    Expands %I to a file name only.
%~xI                    Expands %I to a file extension only.
%~sI                    Expands path to contain short names only.
%~aI                    Expands %I to the file attributes of file.
%~tI                    Expands %I to the date and time of file.
%~zI                    Expands %I to the size of file.
%~$PATH:I               Searches the directories listed in the PATH environment
                        variable and expands %I to the fully qualified name of
                        the first one found. If the environment variable name is
                        not defined or the file is not found by the search,
                        this modifier expands to the empty string.
```

## String operations (BASH)

```
${#string}                  Length of $string

${string:position}          Extract substring from $string at $position
${string:position:length}   Extract $length characters substring from $string at $position

${string#substring}         Strip shortest match of $substring from front of $string
${string##substring}        Strip longest match of $substring from front of $string
${string%substring}         Strip shortest match of $substring from back of $string
${string%%substring}        Strip longest match of $substring from back of $string

${string/substring/replacement}     Replace first match of $substring with $replacement
${string//substring/replacement}    Replace all matches of $substring with $replacement
${string/#substring/replacement}    If $substring matches front end of $string, substitute $replacement
${string/%substring/replacement}    If $substring matches back end of $string, substitute $replacement
```

See “Advanced Bash-Scripting Guide (tldp.org)” for additional information.

# End

## Homework 1/2 - Loop again

Use the loop examples to:

- Count from 6 to 23 (and output those numbers)
- Call FFmpeg to rewrap all files in a folder to Matroska (.mkv)
- Modify that script to accept the folder per Drag-and-Drop on the Batchfile

**Hint:** Drag-and-Drop stuff on “loop_examples.bat”

## Homework 2/2 - Prepare Cygwin

- Download the 64bit Cygwin installer:
  “setup-x86_64.exe”
- Don’t install it yet (we’ll do that together).