BASH

BASH naming limitations

- ◆ Forbidden character in UNIX names
 - / slash = namespace separator
 - <NewLine> = display separator



- ◆ BASH special characters <>|&;*?()"'`\\${}[]
- A
- ♦ + Separators <space>, <TAB> and <NewLine>
 - All can be escaped with \
 - Or deactivated inside quotes '...' excluding '
 - Or deactivated inside dbl-quotes "..." excluding "`\\$
 - Possible file names:





→ BEWARE: file & folder names with BASH characters may cause bugs in « unaware » scripts

BASH Pattern matching for names

- ◆ Generalize names with metacharacters
 - * Zero or any number of any character
 - ? One of any character
 - [class] One character in the group i.e [Aez12]
 - [^class] One character NOT in the group [^ezEZ]
 - [x-y] One character in the interval [0-9]
 - Escape for literal usage of a metacharacter *
- ♦ Works with 1s, rm, mv, cp, etc.

BASH Commands on multiple lines

- ◆ Type (md AA
 - Will not execute command
 - Asks for more information until we type)
 - May be nested
- ◆ To split a command on multiple lines
 - Escape <EOL> with \
 - Type (mkdir \ ← beware the space separator
 - Type AA \ ← beware the space separator
 - Type **BB**

BASH

◆ Lower case

• Type cc) ← will create folder "BBCC"

BASH

Use variables

- ◆ Variables and special characters
 - VAR1="my variable"
 - \$VAR1
- → my variable
- "\$VAR1"
- → my variable → \$VAR1
- '\$VAR1'\${VAR1}
- → my variable
- {\$VAR1}
- → {my variable}
- Variable substitution
 - VAR2="VAR1"
 - \${!VAR2}
- → my variable
- \${VAR3:-"none"} → none
- \${VAR3:-\$VAR1} → my variable

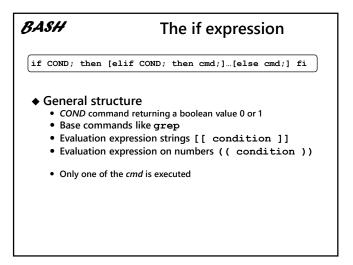
◆ String variable length \${#VAR} ♦ Substring substitution \${VAR/pattern/replace-string} \${VAR#*delete-pattern} Prefix remove to first pattern ◆ Prefix remove to last pattern \${VAR##*delete-pattern} ◆ Suffix remove from last pattern \${VAR%delete-Pattern*} Suffix remove from first pattern \${VAR%%delete-pattern*} Reduction to end \${VAR:offset} ◆ Reduction of length \${VAR:offset:length} \${VAR^^} ◆ Upper case

\${VAR,,}

Variable operations

BASH	Declare & use Arrays	
◆ Declare	declare -a TABL=(AAA BB CCC)	
♦ Use index • echo \${TAI	echo \${TABL[1]} 3L[@]:1:2}	→ BB→ BB CCC
◆ Serialize	echo \${TABL[@]}	→ AAA BB CCC
◆ Cardinal	echo \${#TABL[@]}	→ 3
◆ Element size	echo \${#TABL[1]}	→ 2
◆ Clear element	unset \${TABL[1]}	

BASH **Brace expansion** Special brace expression \rightarrow a b c d e f ... x y z • {a..z} • {0..1} → 0 1 2 3 4 5 6 7 8 9 • {\$from..\$to} ◆ Array construction • ALPHA=({A..Z}) • \${#ALPHA[@]} **→** 26 ♦ String generation • DIGIT=({0..9}) • TYPO=(\${ALPHA[@]} \${DIGIT[@]}) • PASS=\$(shuf -n8 -er \${TYPO[@]} | paste -sd "")



BASH **Extended string operators** ◆ Use in [[...]] expression **♦** String Comparison • Operators ==, !=, < and > • Give a result based on alphabetic order ◆ Generic comparison (use in [[...]]) • -eq • -ne != • -1t < • -le • -gt > • -ge - Operands are strings → alphabetic order - Operands conform with numbers → numeric order

```
#!/bin/bash
INDX=1
while ((INDX < 10 )); do
echo $(INDX)
INDX=$((INDX+1))
done

Useful for infinite loop with exit
• while true; do ...
• End with break, or exit
```

```
BASH
                       The for keyword
 for each-item in collection; do command done
 ◆ Sample script
    #!/bin/bash
    for i in aa bb cc; do
      echo ${i}
    done
    echo ${i}
    done
    for i in {1..5}; do range
      echo ${i}
    for (( i=0; i <= 5; i++ )); do ______C style
      echo ${i}
    for i in {0..7..2}; do ______min, max, increment
```