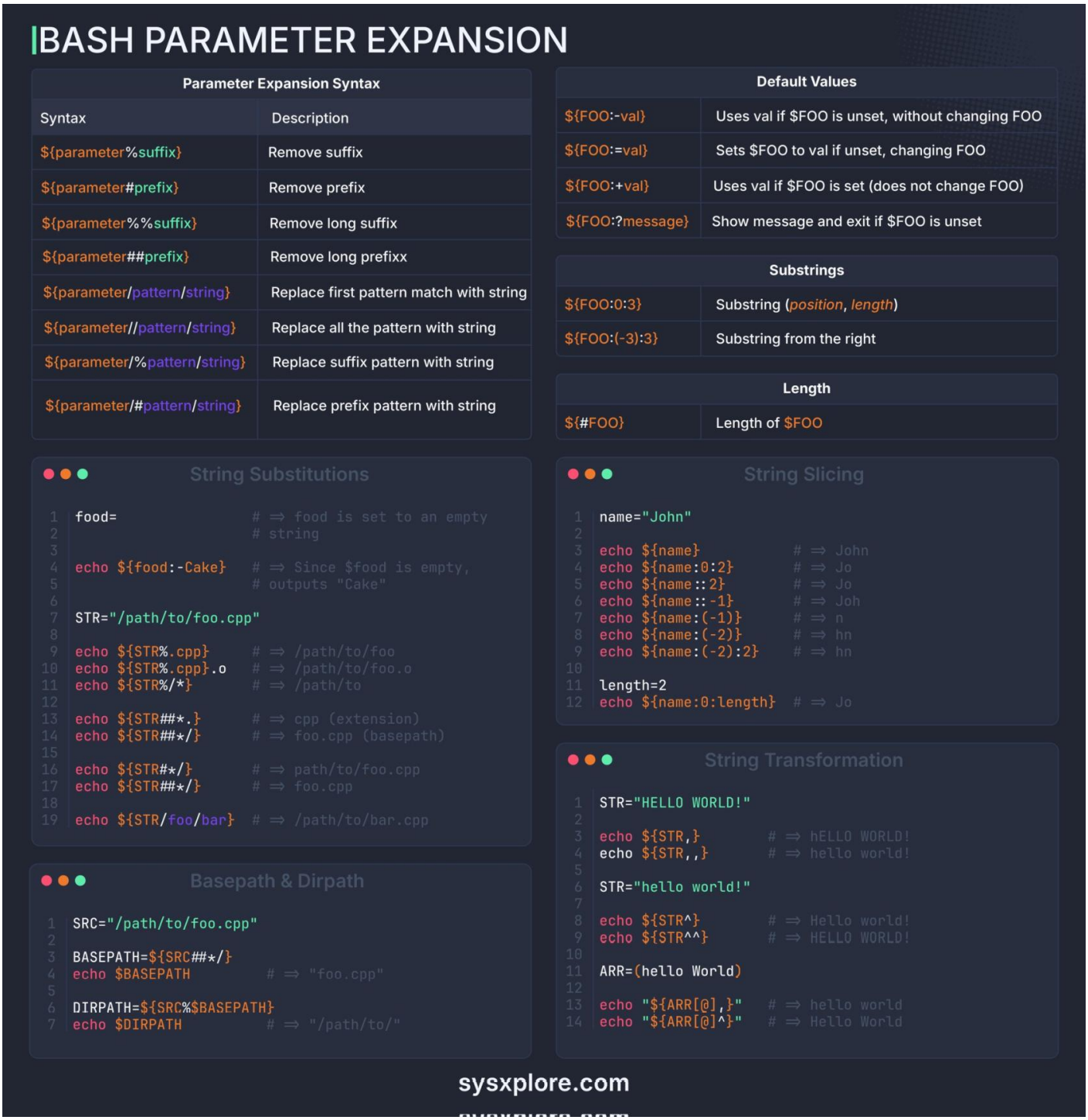
Cheat Sheets

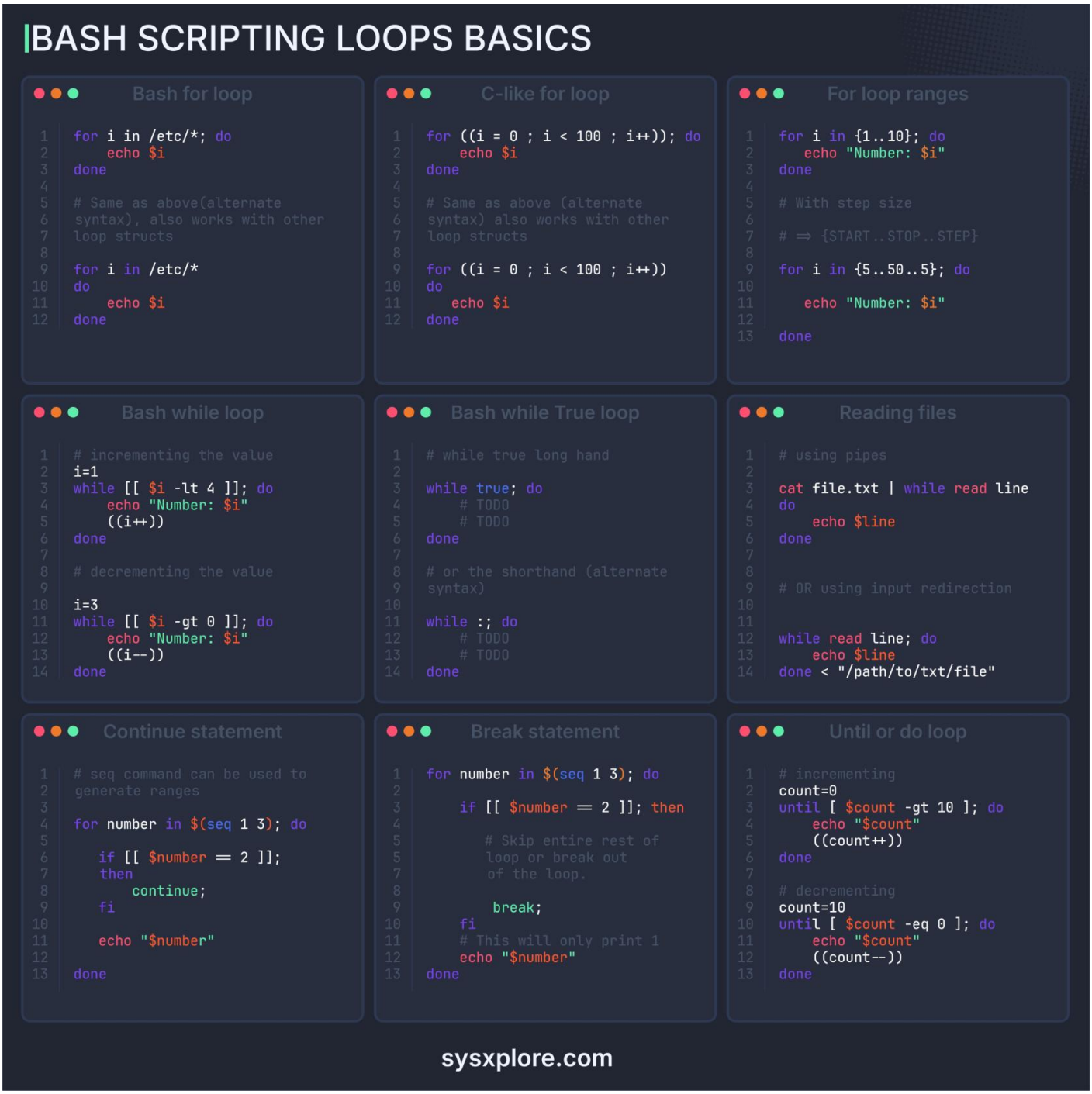
PDF

- bash_cheat_sheet.pdf
- bash-shell-scripting.pdf

Bash Parameter


BASH PARAMETER EXPANSION
Parameter Expansion Syntax
Syntax
Description
${parameter%suffix}
Remove suffix
${parameter#prefix}
Remove prefix
${parameter%%suffix}
Remove long suffix
${parameter##prefix}
Remove long prefixx
${parameter/pattern/string}
Replace first pattern match with string
${parameter//pattern/string}
Replace all the pattern with string
${parameter/%pattern/string}
Replace suffix pattern with string
${parameter/#pattern/string}
Replace prefix pattern with string
Default Values
${FOO:-val}
Uses val if $FOO is unset, without changing FOO
${FOO:=val}
Sets $FOO to val if unset, changing FOO
${FOO:+val}
Uses val if $FOO is set (does not change FOO)
${FOO:?message}
Show message and exit if $FOO is unset
Substrings
${FOO:0:3}
Substring (position, length)
${FOO:(-3):3}
Substring from the right
Length
${#FOO}
Length of $FOO
String Substitutions
food= # ⇒ food is set to an empty
# string
echo ${food:-Cake} # ⇒ Since $food is empty,
# outputs "Cake"
STR="/path/to/foo.cpp"
echo ${STR%.cpp} # ⇒ /path/to/foo
echo ${STR%.cpp}.o # ⇒ /path/to/foo.o
echo ${STR%/*} # ⇒ /path/to
echo ${STR##*.} # ⇒ cpp (extension)
echo ${STR##*/} # ⇒ foo.cpp (basepath)
echo ${STR#*/} # ⇒ path/to/foo.cpp
echo ${STR##*/} # ⇒ foo.cpp
echo ${STR/foo/bar} # ⇒ /path/to/bar.cpp
String Slicing
name="John"
echo ${name} # ⇒ John
echo ${name:0:2} # ⇒ Jo
echo ${name::2} # ⇒ Jo
echo ${name::-1} # ⇒ Joh
echo ${name:(-1)} # ⇒ n
echo ${name:(-2)} # ⇒ hn
echo ${name:(-2):2} # ⇒ hn
length=2
echo ${name:0:length} # ⇒ Jo
String Transformation
STR="HELLO WORLD!"
echo ${STR,} # ⇒ hELLO WORLD!
echo ${STR,,} # ⇒ hello world!
STR="hello world!"
echo ${STR^} # ⇒ Hello world!
echo ${STR^^} # ⇒ HELLO WORLD!
ARR=(hello World)
echo "${ARR[@],}" # ⇒ hello world
echo "${ARR[@]^}" # ⇒ Hello World
Basepath & Dirpath
SRC="/path/to/foo.cpp"
BASEPATH=${SRC##*/}
echo $BASEPATH # ⇒ "foo.cpp"
DIRPATH=${SRC%$BASEPATH}
echo $DIRPATH # ⇒ "/path/to/"
sysxplore.com

Bash Loops


BASH SCRIPTING LOOPS BASICS
Bash for loop
for i in /etc/*; do
echo $i
done
Same as above(alternate syntax), also works with other loop structs
for i in /etc/*
do
echo $i
done
C-like for loop
for ((i = 0 ; i < 100 ; i++)); do
echo $i
done
# Same as above (alternate syntax) also works with other loop structs
for ((i = 0 ; i < 100 ; i++))
do
echo $i
done
For loop ranges
for i in {1..10}; do
echo "Number: $i"
done
# With step size
# ⇒ {START..STOP..STEP}
for i in {5..50..5}; do
echo "Number: $i"
done
Bash while loop
# incrementing the value
i=1
while [[$i -lt 4]]; do
echo "Number: $i"
((i++))
done
# decrementing the value
i=3
while [[$i -gt 0]]; do
echo "Number: $i"
((i--))
done
Bash while True loop
# while true long hand
while true; do
# TODO
# TODO
done
# or the shorthand (alternate syntax)
while :; do
# TODO
# TODO
done
Reading files
using pipes
cat file.txt | while read line
do
echo $line
done
# OR using input redirection
while read line; do
echo $line
done < "/path/to/txt/file"
Continue statement
# seq command can be used to generate ranges
for number in $(seq 1 3); do
if [[$number == 2]];
then
continue;
fi
echo "$number"
done
Break statement
for number in $(seq 1 3); do
if [[$number == 2]]; then
# Skip entire rest of loop or break out of the loop.
break;
fi
This will only print 1
echo "$number"
done
Until or do loop
# incrementing
count=0
until [$count -gt 10]; do
echo "$count"
((count++))
done
# decrementing
count=10
until [$count -eq 0]; do
echo "$count"
((count--))
done
sysxplore.com

Bash Basics

Bash Brackets (??)

Bash Functions

Revision #6
Created 2024-11-11 19:13:23 CST by A-Lang (Admin)
Updated 2025-07-18 19:41:39 CST by A-Lang (Admin)