advanced bash scripting

christoph.gysin@ericsson.com

introduction

```
$ echo "Hello, world!"
```

```
Hello, world!
```

variables

```
$ foo=bar
```

```
$ foo=bar
```

```
$ echo ${foo}
```

```
bar
```

variables

```
$ message="Hello"
```

```
$ echo ${message}
```

```
Hello
```

```
$ message+=", world!"
```

```
$ echo ${message}
```

```
Hello, world!
```

variables

```
$ message="x      x"
```

```
$ echo ${message}
```

```
x x
```

```
$ echo "${message}"
```

```
x      x
```

variables

```
$ string="abcdefg"
```

```
$ echo ${string:3}
```

```
defg
```

```
$ echo ${string:4:2}
```

```
ef
```

```
$ echo ${string: -1}
```

```
g
```

quoting

```
$ string="abcdefg"
```

```
$ echo "\$string"
```

```
$string
```

```
$ echo '$string'
```

```
$string
```

quoting

```
$ msg=$'Hello,\nWorld!'
```

```
$ echo "${msg}"
```

```
Hello,
World!
```

```
$ echo $'\U1f60e'
```

```
😎
```

translation

```
$ echo $"Hello, World!"
```

```
Hello, World!
```

unsafe, discouraged!

special variables

```
$ echo ${USER}
```

```
chris
```

```
$ echo ${BASH_VERSION}
```

```
4.3.27(1)-release
```

```
$ echo $$
```

```
1568
```

special variables

```
$ echo $?
```

```
0
```

```
$ false
```

```
$ echo $?
```

```
1
```

special variables

```
$ echo $0
```

```
/bin/bash
```

```
$ echo $1
```

```

```

special variables

```
$ bash -c 'echo $1' foo bar baz
```

```
bar
```

```
$ bash -c 'echo $@' foo bar baz
```

```
bar baz
```

```
$ bash -c 'echo $0 $@' foo bar baz
```

```
foo bar baz
```

special variables

```
$ bash -c 'echo $#' foo bar baz
```

```
2
```

arrays

```
$ a=(abc def)
```

```
$ echo ${a}
```

```
abc
```

```
$ echo ${a[1]}
```

```
def
```

```
$ echo ${a[@]}
```

```
abc def
```

arrays

```
$ a=(a b c)
```

```
$ a+=(d e f)
```

```
$ echo "${a[@]}"
```

```
a b c d e f
```

arrays

```
$ a=(a b c)
```

```
$ a+=d
```

```
$ echo "${a[@]}"
```

```
ad b c
```

```
$ a+=(d)
```

```
$ echo "${a[@]}"
```

```
$ ad b c d
```

arrays

```
$ a=(a b c d e f g)
```

```
$ echo ${#a[@]}
```

```
7
```

```
$ echo ${a[@]:2}
```

```
c d e f g
```

```
$ echo ${a[@]:2:3}
```

```
c d e
```

associative arrays

```
$ unset a
```

```
$ declare -A a
```

```
$ a=([name]=bob [age]=21)
```

```
$ echo ${a[name]}
```

```
bob
```

```
$ echo ${a[age]}
```

```
21
```

associative arrays

```
$ a[age]=23
```

```
$ a[email]=bob@domain.tld
```

```
$ echo ${a[email]}
```

```
bob@domain.tld
```

integer

```
$ declare -i int
```

```
$ int=foo
```

```
$ echo ${int}
```

```
0
```

```
int=2+2
```

```
$ echo ${int}
```

```
4
```

uppercase

```
$ declare -u upper
```

```
$ upper="Hello, World!"
```

```
$ echo ${upper}
```

```
HELLO, WORLD!
```

functions

```
$ function greeter() { echo "Hello ${USER}!"; }
```

```
$ greeter
```

```
Hello chris!
```

```
$ function greeter() { echo "Hello $1!"; }
```

```
$ greeter bob
```

```
Hello bob!
```

local variables

```
$ foo=1
```

```
$ function f() {
    foo=2
}
```

```
$ f
```

```
$ echo ${foo}
```

```
2
```

local variables

```
$ function f() {
    echo before: ${foo}
    local foo=3
    echo after: ${foo}
}
```

```
$ f
```

```
before: 2
after: 3
```

```
$ echo ${foo}
```

```
2
```

lists

```
$ echo foo; echo bar
```

```
foo
bar
```

```
$ true && echo "it's true!"
```

```
it's true!
```

```
$ false && echo "is it?" || echo "nope, that's false"
```

```
nope, that's false
```

test

```
$ test foo = foo; echo $?
```

```
0
```

```
$ test foo = foo && echo true || echo false
```

```
true
```

```
$ test foo = bar && echo true || echo false
```

```
false
```

test

```
$ test foo = bar
```

```
$ [ foo = bar ]
```

```
$ [ foo = bar ] && echo true || echo false
```

```
false
```

test

```
[ ! <expression> ] # expression is false
```

```
[ <expr1> -a <expr2> ] # both are true
```

```
[ <expr1> -o <expr2> ] # either is true
```

```
[ -n <string> ] # string length is nonzero
```

```
[ -z <string> ] # string length is zero
```

```
[ <string1> = <string2> ] # strings are equal
```

```
[ <string1> != <string2> ] # strings are not equal
```

test

```
[ <int1> -eq <int2> ] # integers are equal
```

```
[ <int1> -ne <int2> ] # integers are not equal
```

```
[ <int1> -lt <int2> ] # int1 is less than int2
```

```
[ <int1> -le <int2> ] # int1 is less or equal int2
```

```
[ <int1> -gt <int2> ] # int1 is greater than int2
```

```
[ <int1> -ge <int2> ] # int1 is greater or equal int2
```

test

```
[ -e <file> ] # file exists
```

```
[ -f <file> ] # file exists and is a regular file
```

```
[ -L <file> ] # file exists and is a symbolic link
```

```
[ -d <file> ] # file exists and is a directory
```

```
[ -s <file> ] # file exists and has size greater than 0
```

```
[ -r <file> ] # file exists and is readable
```

```
[ -x <file> ] # file exists and is executable
```

control structures: if/elif/else/fi

```
$ function f() {
    if [ $1 -lt 0 ]; then
        echo "negative"
    elif [ $1 -le 9 ]; then
        echo "one digit"
    elif [ $1 -le 99 ]; then
        echo "two digits"
    else
        echo "large!"
    fi
}
```

```
$ f 5
```

```
one digit
```

control structures: if/elif/else/fi

```
$ function f() {
    if [ $1 -lt 0 ]; then
        echo "negative"
    elif [ $1 -le 9 ]; then
        echo "one digit"
    elif [ $1 -le 99 ]; then
        echo "two digits"
    else
        echo "large!"
    fi
}
```

```
$ f -1
```

```
negative
```

control structures: if/elif/else/fi

```
$ function f() {
    if [ $1 -lt 0 ]; then
        echo "negative"
    elif [ $1 -le 9 ]; then
        echo "one digit"
    elif [ $1 -le 99 ]; then
        echo "two digits"
    else
        echo "large!"
    fi
}
```

```
$ f 2134567890
```

```
large!
```

control structures: while/do/done

```
$ function f() {
    local -i i=0
    local args=($@)

    while [ ${i} -lt $# ]; do
        echo "${i}: ${args[${i}]}"
        i=i+1
    done
}
```

```
$ f foo bar baz
```

```
0: foo
1: bar
2: baz
```

control structures: case/in/esac

```
$ function nth() {
    case ${1: -1} in
        1)      echo "${1}st" ;;
        2)      echo "${1}nd" ;;
        3)      echo "${1}rd" ;;
        [4-9]) echo "${1}th" ;;
        *)      echo "not a number" ;;
    esac
}
```

```
$ nth 3
```

```
3rd
```

control structures: case/in/esac

```
$ function nth() {
    case ${1: -1} in
        1)      echo "${1}st" ;;
        2)      echo "${1}nd" ;;
        3)      echo "${1}rd" ;;
        [4-9]) echo "${1}th" ;;
        *)      echo "not a number" ;;
    esac
}
```

```
$ nth 105
```

```
105th
```

control structures: case/in/esac

```
$ function nth() {
    case ${1: -1} in
        1)      echo "${1}st" ;;
        2)      echo "${1}nd" ;;
        3)      echo "${1}rd" ;;
        [4-9]) echo "${1}th" ;;
        *)      echo "not a number" ;;
    esac
}
```

```
$ nth foo
```

```
not a number
```

control structures: for/do/done

```
$ for((i=1; i<=5; ++i)); do
    echo ${i}
done
```

```
1
2
3
4
5
```

control structures: for/in/do/done

```
$ for number in 1 2 3 4 5; do
    echo ${number}
done
```

```
1
2
3
4
5
```

control structures: for/in/do/done

```
$ names=("Alice" "Bob jr." "Carol" "Dave")
```

```
$ for name in ${names[@]}; do # careful!
    echo ${name}
done
```

```
Alice
Bob
jr.
Carol
Dave
```

control structures: for/in/do/done

```
$ names=("Alice" "Bob jr." "Carol" "Dave")
```

```
$ for name in "${names[@]}"; do
    echo ${name}
done
```

```
Alice
Bob jr.
Carol
Dave
```

command substitution

```
$ date +%s
```

```
1412160589
```

```
$ seconds=$(date +%s)
```

```
$ echo ${seconds}
```

```
1412160713
```

command substitution

```
$ cat /etc/hostname
```

```
echrgys-laptop
```

```
$ hostname=$(cat /etc/hostname) # slow!
```

```
$ hostname=$(</etc/hostname)
```

```
$ echo ${hostname}
```

```
echrgys-laptop
```

compound commands: group

```
$ { echo a; echo b }
```

```
a
b
```

```
$ f=file.txt
$ [ -f ${f} ] || { echo "creating file"; touch ${f}; }
```

compound commands: subshell

```
$ pwd
/home/chris
```

```
$ (cd /tmp; touch temp.txt)
```

```
$ pwd
/home/chris
```

```
$ ls /tmp/*.txt
/tmp/temp.txt
```

compound commands: subshell

```
$ i=1
```

```
$ (i=2)
```

```
$ echo $i
```

```
1
```

pipelines

```
$ (echo abc; echo def; echo ghi)
```

```
abc
def
ghi
```

```
$ (echo abc; echo def; echo ghi) | tac
```

```
ghi
def
abc
```

pipelines

```
$ (echo abc; echo def; echo ghi) | tac | rev
```

```
ihg
fed
cba
```

```
$ (echo abc; echo def; echo ghi) | tac | rev | tail -n1
```

```
cba
```

pipelines

```
$ ssh esekilxxen981 getent passwd |
    grep ^lmf | cut -d: -f5 | sort | head
```

```
Aarne Nurmi
Aila Koponen
Andras Vajda
Ann-Mari Karjala
Anna Sippel
Annamari Laurikainen
Anneli Granstrom
Antero Vanska
Antti Alinen
Ari Greus
```

sequence expression

```
$ echo {1..10}
```

```
1 2 3 4 5 6 7 8 9 10
```

```
$ echo {0..100..7}
```

```
0 7 14 21 28 35 42 49 56 63 70 77 84 91 98
```

```
$ echo {00..999}
```

```
000 001 002 003 004 005 006 007
008 009 010 011 012 013 014 015
[...]
992 993 994 995 996 997 998 999
```

brace expansion

```
$ echo {bgf,mrfc,mrfp}_appl
```

```
bgf_appl mrfc_appl mrfp_appl
```

```
$ echo {bgf,mrf{c,p}}_appl
```

```
bgf_appl mrfc_appl mrfp_appl
```

brace-expansion

```
$ cp source.cpp{,.bak}
```

```
$ mkdir -p /mnt/really/long/path/with/typo/is/annoiing
```

```
$ mv /mnt/really/long/path/with/typo/is/annoiing \
     /mnt/really/long/path/with/typo/is/annoying
```

```
$ cd /mnt/really/long/path/with/typo/is
$ mv annoiing annoying
$ cd ~
```

```
$ (cd /mnt/really/long/path/with/typo/is; \
   mv annoiing annoying)
```

brace expansion

```
$ mv /mnt/really/long/path/with/typo/is/anno{i,y}ing
```

parameter expansion

```
$ echo ${s:-"default"}
```

```
default
```

```
$ s=custom
```

```
$ echo ${s:-"default"}
```

```
custom
```

parameter expansion

```
$ unset s
```

```
$ echo ${s:="assign"}
```

```
assign
```

```
$ echo ${s}
```

```
assign
```

parameter expansion

```
$ unset s
```

```
$ echo s is: ${s:+"alternate"}
```

```
s is:
```

```
$ s=custom
```

```
$ echo s is: ${s:+"alternate"}
```

```
s is: alternate
```

parameter expansion

```
$ s="a.b.c.d.e"
```

```
$ echo ${s%.e}
```

```
a.b.c.d
```

```
$ echo ${s%.*}
```

```
a.b.c.d
```

```
$ echo ${s%%.*}
```

```
a
```

parameter expansion

```
$ s="a.b.c.d.e"
```

```
$ echo ${s#a.}
```

```
b.c.d.e
```

```
$ echo ${s#*.}
```

```
b.c.d.e
```

```
$ echo ${s##*.}
```

```
e
```

parameter expansion

```
$ f=/some/path/to/file.txt
```

```
$ basename ${f}
```

```
file.txt
```

```
$ echo ${f##*/}
```

```
file.txt
```

parameter expansion

```
$ f=/some/path/to/file.txt
```

```
$ dirname ${f}
```

```
/some/path/to
```

```
$ echo ${f%/*}
```

```
/some/path/to
```

parameter expansion

```
$ unset s
```

```
$ echo s is: ${s:-"default"}
```

```
s is: default
```

```
$ s=""
```

```
$ echo s is: ${s:-"default"}
```

```
s is: default
```

parameter expansion

```
$ unset s
```

```
$ echo s is: ${s-"default"}
```

```
s is: default
```

```
$ s=""
```

```
$ echo s is: ${s-"default"}
```

```
s is:
```

regular expressions

```
$ [[ "bar" =~ ^ba ]]
```

```
$ for s in foo bar baz; do
    if [[ "${s}" =~ ^ba ]]; then
        echo match: ${s}
    fi
done
```

```
match: bar
match: baz
```

I/O redirection

```
$ echo foo > file.txt
```

```
$ cat file.txt
```

```
foo
```

```
$ echo bar > file.txt
```

```
$ cat file.txt
```

```
bar
```

I/O redirection

```
$ echo foo > file.txt
```

```
$ echo bar >> file.txt
```

```
$ cat file.txt
```

```
foo
bar
```

I/O redirection

```
$ function log() { echo >&2 "$@"; }
```

```
$ log "This is an error message!"
```

```
This is an error message!
```

```
$ log "error" > error.log
```

```
error
```

```
$ [ -s error.log ] || echo empty
```

```
empty
```

I/O redirection

```
$ log "error" 2> error.log
```

```
$ cat error.log
```

```
error
```

I/O redirection

```
$ strace ls
```

```
execve("/usr/bin/ls", ["ls"], [/* 70 vars */]) = 0
brk(0)                                  = 0xd04000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[...]
```

```
$ strace ls | grep TIOCGWINSZ
```

```
execve("/usr/bin/ls", ["ls"], [/* 70 vars */]) = 0
brk(0)                                  = 0xd04000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[...]
```

I/O redirection

```
$ strace ls 2>&1 | grep TIOCGWINSZ
```

```
ioctl(1, TIOCGWINSZ, 0x7fffb02fa7a0)    = -1 ENOTTY
```

```
$ strace ls |& grep TIOCGWINSZ
```

```
ioctl(1, TIOCGWINSZ, 0x7fffb02fa7a0)    = -1 ENOTTY
```

I/O redirection

```
$ cut -d: -f1 < /etc/passwd
```

```
root
bin
daemon
mail
[...]
```

exec

```
$ bash -c 'id -un; echo foo'
```

```
chris
foo
```

```
$ bash -c 'exec id -un; echo foo'
```

```
chris
```

exec

```
bash -c 'echo $$; readlink /proc/self'
```

```
11726
11727
```

```
bash -c 'echo $$; exec readlink /proc/self'
```

```
11728
11728
```

exec (redirection)

```
$ exec {FD}> file.txt
```

```
$ echo ${FD}
```

```
10
```

```
$ echo >&${FD} foobar
```

```
$ cat file.txt
```

```
foobar
```

```
$ exec ${FD}>&-
```

exec (redirection)

```
$ exec 666> file.txt
```

```
$ ls -l /dev/fd/666
```

```
/dev/fd/666 -> /home/chris/file.txt
```

```
$ echo >&666 foobar
```

```
$ cat file.txt
```

```
foobar
```

```
$ exec 666>&-
```

useless use of cat

```
$ cat /etc/passwd | grep ${USER}
```

```
$ grep ${USER} < /etc/passwd
```

```
$ grep ${USER} /etc/passwd
```

process substitution

```
$ cat <(echo foo)
```

```
foo
```

```
$ echo <(echo foo)
```

```
/dev/fd/63
```

process substitution

```
$ ssh esekilx5160 rpm -qa | sort > 5160.pkgs
```

```
$ ssh esekilx5163 rpm -qa | sort > 5163.pkgs
```

```
$ diff -u 516{0,3}.pkgs
```

```
$ rm 516{0,3}.pkgs
```

process substitution

```
$ diff -u <(ssh esekilx5160 rpm -qa | sort) \
          <(ssh esekilx5163 rpm -qa | sort)
```

```
--- /dev/fd/63  2014-10-02 09:36:27.445762659 +0300
+++ /dev/fd/62  2014-10-02 09:36:27.445762659 +0300
@@ -573,7 +573,11 @@
 gnome-pilot-2.0.16-3.44.68
 gnome-pilot-devel-2.0.16-3.44.68
 gnome-pilot-lang-2.0.16-3.44.68
+gnome-power-manager-2.24.1-17.67.1
+gnome-power-manager-lang-2.24.1-17.67.1
 gnome-print-sharp-2.26.0-2.2.7
+gnome-screensaver-2.28.3-0.32.1
[...]
```

job control

```
$ (sleep $[20*60]; echo "wake up!")
```

```
wake up!
```

```
$ (sleep $[20*60]; echo "wake up!") &
```

```
[1] 12019
```

```
$ kill %1
```

```
[1]+  Terminated    ( sleep $[20*60]; echo "wake up!" )
```

trap

```
$ bash -c 'trap "echo Ctrl-C pressed!" TERM; sleep 60'
```

```
^C
```

```
Ctrl-C pressed!
```

```
$ bash -c 'trap "echo exiting..." EXIT; sleep 60' &
```

```
[1] 12164
```

```
$ kill %1
```

```
exiting...
```

getopts

- builtin
- simple syntax
- no long options
- argument order matters

discouraged, use getopt instead!

getopt

```
OPT=$(
    getopt \
        --options hv: \
        --long help \
        --long value: \
        --name "$0" \
        -- "$@"
)

if [ $? -ne 0 ]; then
    usage
    exit 1
fi
```

getopt

```
eval set -- "${OPT}"

while true; do
    case "$1" in
        -h|--help)
            usage
            exit 0
            ;;
        -v|--value)
            value=$2
            shift 2
            ;;
    esac
done
```

eval

```
$ var1=foo
```

```
$ var2=bar
```

```
$ declare -i i=1
```

```
$ echo ${var${i}}
```

```
bash: ${var${i}}: bad substitution
```

eval

```
$ var1=foo
```

```
$ var2=bar
```

```
$ declare -i i=1
```

```
$ eval "echo \${var${i}}"
```

```
foo
```

eval

```
$ var1=foo
```

```
$ var2=bar
```

```
$ declare -i i=1
```

```
$ i=i+1
```

```
$ eval "echo \${var${i}}"
```

```
bar
```

good habits

```
#!/bin/bash
```

```
$ bash -c "false"; echo $?
```

```
1
```

```
$ bash -c "false; echo done"; echo $?
```

```
done
0
```

good habits

```
#!/bin/bash
set -e
```

```
$ bash -c "set -e; false; echo done"; echo $?
```

```
1
```

good habits

```
#!/bin/bash
set -e
```

```
$ bash -c 'message="foo"; echo msg=${mesage}'
```

```
msg=
```

good habits

```
#!/bin/bash
set -e
set -u
```

```
$ bash -c 'set -u; message="foo"; echo msg=${mesage}'
```

```
bash: mesage: unbound variable
```

pitfalls

```
$ cat file
```

```
foo
bar
```

```
$ x="start"
```

```
$ cat file | while read i; do x+=",$i"; done; echo $x
```

```
start
```

pitfalls

```
$ cat file
```

```
foo
bar
```

```
$ x="start"
```

```
$ while read i; do x+=",$i"; done < file; echo $x
```

```
start,foo,bar
```

pitfalls

```
$ set -e
$ func() {
    local var=$(false)
    echo "done"
}
```

```
$ func
```

```
done
```

pitfalls

```
$ set -e
$ func() {
    local var
    var=$(false)
    echo "done"
}
```

```
$ func
```

```
(shell exits with code 1)
```

coding style

```
google-styleguide.googlecode.com/svn/trunk/shell.xml
```

shellcheck

```
https://github.com/koalaman/shellcheck
```

```
$ shellcheck /usr/bin/gettext.sh
```

```
In /usr/bin/gettext.sh line 20:
if test "X`(echo '\t') 2>/dev/null" = 'X\t'; then
          ^-- SC2006: Use $(..) instead of `..
                ^-- SC2028: echo won't expand escape
                            sequences. Consider printf.
```

the end

questions?