1. Командний інтерпретатор bash

NAME

bash - GNU Bourne-Again SHell

SYNOPSIS

bash [options] [command_string | file]

DESCRIPTION

Командний інтерпретатор *bash* (далі – КІ bash) здійснює виконання команд, які надходять з потоку стандартного введення або файлу. КІ bash включає функціональні можливості з ksh і csh, призначений для сумісної реалізації командної оболонки і набору утиліт CLI згідно IEEE POSIX specification (IEEE Standard1003.1). За вимогами конфігурування POSIX-сумісності КІ bash обирається за замовчанням.

OPTIONS

-c читає і виконує командний рядок (КР), який є першим аргументом ***command_string*** після даної опції або списку всіх опцій. Вміст командного рядка може бути одержаний як значення відповідного позиційного параметру;

-i вмикає інтерактивний режим виконнаня КР;

-- опції КР після даного символу не обробляються.

MULTI-CHARACTER OPTIONS

--help вивести стислу довідку з використання і завершити роботу;

--init-file file аргументом опції є file, якій підлягає обробленню;

--rcfile file аргументом опції є file, якій буде ***виконаний*** замість загальносистемного файлу ініціалізації /etc/bash.bashrc і персонального ~/.bashrc, якщо КІ bash працює в інтерактивному режимі (див. INVOCATION);

--noprifile не читає вміст загальносистемного файлу ініціалізації або будь-яких персональних /etc/profile, ~/.bash_profile, ~/.bash_login або ~/.profile. За замовчанням КІ bash читає ці файли (див. INVOCATION);

--norc не читає і не виконує вміст загальносистемного файлу /etc/bash.bashrc і персонального ~/.bashrc, якщо КІ bash працює в інтерактивному режимі;

--verbose режим більшої інформативності;

--version вивести версію програмної реалізації і завершити роботу.

DEFINITIONS

blank пробіл або знак табуляції;

word цілісна послідовність символів або токен;

name буквено-числовий ідентифікатор;

metacharacter **| & ; ( ) < > space tab**

control operator **|| & && ; ;; ( ) | |& <newline>**

RESERVED WORDS

Зарезервовані слова мають спеціальні значення:

! case coproc do done elif else esac fi for function if in select then until while { } time [[ ]]

SHELL GRAMMAR

Simple Commands

Прості команди є змінною послідовністю слів (words) через пробіл і символ перенаправлення (redirections), яка завершується оператором управління (control operator)

Pipelines

Конвеєр (pipeline) є послідовністю кількох команд (commands) об'єднаних через оператори управління (control operators) **|** або **|&**

Lists

Список (list) є послідовністю кількох конвеєрів (pipelines) об'єднаних через оператори **;**, **&**, &&, або **||**, і іноді завершуються на один з символів **;**, **&** або **<newline>**

command1 **&&** command2

command2 буде виконана тоді, коли **command1** буде виконана успішно

command1 **||** command2

command2 буде виконана тоді, коли **command1** завершиться успішно

SHELL GRAMMAR (продовження)

Compound Commands

(list) **{ list; }** **((expression))** **[[expression]]**

(expression) **!expression**
expression1 && expression2 **expression1 || expression2**

for name [[**in** [word ...]] ;] **do** list ; **done**

for ((expr1 ; expr2 ; expr3)) ; **do** list ; **done**

select name [**in** word] ; **do** list ; **done**

case word **in** [[(] pattern [| pattern] ...) list ;;] ... **esac**

if list; **then** list; [**elif** list; **then** list;] ... [**else** list;] **fi**

while list-1; **do** list-2; **done**

until list-1; d**o** list-2; **done**

SHELL BUILTIN COMMANDS

source filename [arguments]

alias [-p] [name[=value] ...]

bg [jobspec ...]

bind [-m keymap] [-lpsvPSVX]

break [n]

builtin shell-builtin [arguments]

caller [expr]

cd [-L|[-P [-e]] [-@]] [dir]

command [-pVv] command [arg ...]

compgen [option] [word]

SHELL BUILTIN COMMANDS (продовження)

compopt [-o option] [-DE] [+o option] [name]

continue [n]

declare [-aAfFgilnrtux] [-p] [name[=value] ...]

typeset [-aAfFgilnrtux] [-p] [name[=value] ...]

dirs [-clpv] [+n] [-n]

disown [-ar] [-h] [jobspec ...]

echo [-neE] [arg ...]

enable [-a] [-dnps] [-f filename] [name ...]

eval [arg ...]

SHELL BUILTIN COMMANDS (продовження)

exec [-cl] [-a name] [command [arguments]]

exit [n]

export [-fn] [name[=word]] ...

fc [-e ename] [-lnr] [first] [last]

fg [jobspec]

getopts optstring name [args]

hash [-lr] [-p filename] [-dt] [name]

help [-dms] [pattern]

history [n]

jobs [-lnprs] [jobspec ...]

SHELL BUILTIN COMMANDS (продовження)

kill [-s sigspec | -n signum | -sigspec] [pid | jobspec]

let arg [arg ...]

local [option] [name[=value] ...]

logout

mapfile [-n count] [-O origin] [-s count] [-t] [-u fd] [-C callback] [-c quantum] [array]

readarray [-n count] [-O origin] [-s count] [-t] [-u fd] [-C callback] [-c quantum] [array]

popd [-n] [+n] [-n]

printf [-v var] format [arguments]

pushd [-n] [+n] [-n]

pwd [-LP]

SHELL BUILTIN COMMANDS (продовження)

read [-ers] [-a aname] [-d delim] [-i text] [-n nchars] [-N nchars] [-p prompt]
[-t timeout] [-u fd] [name ...]

readonly [-aAf] [-p] [name[=word] ...]

return [n]

set [--abefhkmnptuvxBCEHPT] [-o option-name] [arg ...]

shift [n]

shopt [-pqsu] [-o] [optname ...]

suspend [-f]

test expr

times

SHELL BUILTIN COMMANDS (завершення)

trap [-lp] [[arg] sigspec ...]

type [-aftpP] name [name ...]

ulimit [-HSTabcdefilmnpqrstuvx [limit]]

umask [-p] [-S] [mode]

unalias [-a] [name ...]

unset [-fv] [-n] [name ...]

wait [-n] [n ...]

2. Основи програмування мовою командного інтерпретатора bash

1

```
#!/bin/bash

ping -c3 -l3 -w2 192.168.0.3
```

2

```
#!/bin/bash
ping -n -c3 -l3 -w2 $1      #$1 - 1-й позиційний параметр
```

3

```
#!/bin/bash
#Ініціалізація цілі
TARGET=$1      #$1 - 1-й позиційний параметр
#Перевірка наявності цілі
if test -z $TARGET # [ -z $TARGET ] - альтернативний варіант test -z $TARGET
then
   #Запит цілі від користувача
   echo -n "Enter target address: "
   #Читання користувальницького введення
   read TARGET
fi
ping -n -c3 -l3 -w2 $TARGET
```

2. Основи програмування мовою командного інтерпретатора bash

4

```
#!/bin/bash
#Зондування цілей, заданих у командному рядку
while [ $# -gt 0 ]      #$# - кількість позиційних параметрів
do
   ping -n -c3 -l3 -w2 $1  #$1 - 1-й позиційний параметр
   #Виведення порожнього рядка
   echo
   #Зсув позиційних параметрів (на 1 позицію ліворуч)
   shift
done
```

5

```
#!/bin/sh
#grep видаляє порожні рядки та рядки коментарів (рядки, що починаються з #)
cat $HOME/pingall-hosts | grep -E -v "^$|^#" | while read TARGET
do
   #Зондування чергової цілі
   ping -n -c3 -l3 -w2 $TARGET
   #Виведення порожнього рядка
   echo
done
```

2. Основи програмування мовою командного інтерпретатора bash

6

```
#!/bin/bash
if [ ! -z $1 ]
then
   CONFIG_FILE=$1 #Конфигураційний файл заданий в командному рядку
else
   CONFIG_FILE=$HOME/pingall-hosts   # Якщо конфигураційний файл не заданий
fi
#Перевірка існування конфігураційного файлу
if [ ! -f $CONFIG_FILE ]
then    # Конфігураційний файл не існує
   echo "Configuration file $CONFIG_FILE is not exist”
   exit 1
fi
#grep видаляє порожні рядки та рядки коментарів (рядки, що починаються з #)
cat $CONFIG_FILE | grep -E -v "^$|^#" | while read TARGET
do
   ping -n -c3 -l3 -w2 $TARGET; echo
done
```

```
#!/bin/bash
DEFAULT_CONFIG_FILE=$HOME/pingall-hosts
#ТІЛО СЦЕНАРІЯ
case $1 in        #$1 - 1-й позиційний параметр, $2 - 2-й и т. д.
-h) #Користувачу потрібна лише довідка
   echo "usage: `basename $0` [-h | -t target | -c config_file]" >&2
   exit 0
   ;;
-t) #Користувач явно задав ціль
   ping -n -c3 -l3 -w2 $2
   exit 0
   ;;
-c) #Користувач явно вказав конфігураційний файл
   CONFIG_FILE=$2
   ;;
"") #Користувач виконав виклик сценарія без аргументів (використовувати
конфігураційний файл по замовчанню)
   CONFIG_FILE=$DEFAULT_CONFIG_FILE
   ;;
esac
```

2. Основи програмування мовою командного інтерпретатора bash

(продовження)

```
#Перевірка, чи заданий конфігураційний файл
if [ -z $CONFIG_FILE ]
then #Конфігураційний файл не заданий
    echo "Configuration file is not defined" >&2
    exit 1
fi
#Перевірка, чи існує конфігураційний файл
if [ ! -f $CONFIG_FILE ]
then #Конфігураційний файл не існує
    echo "Configuration file $CONFIG_FILE is not exist" >&2
    exit 1
fi

#Зондування цілей, заданих у конфігураційному файлі
#grep видаляє порожні рядки та рядки коментарів (рядки, що починаються з #)
cat $CONFIG_FILE | grep -E -v "^$|^#" | while read TARGET
do
    ping -n -c3 -l3 -w2 $TARGET; echo
done
```

```
#!/bin/sh
CONFIG_FILE=$HOME/pingall-hosts      #Ім'я конфігураційного файлу по замовчанню
TARGET=                              #Ціль
VERBOSE=off                          #Перемикач говірливості
COUNTER=0                            #Лічильник прозондованих вузлів
#ВИЗНАЧЕННЯ ФУНКЦІЙ
pingone()
{
   ping -n -c3 -l3 -w2 $TARGET; echo
}

usage()
#Виводить інформацію про синтаксис
{
   echo "usage: `basename $0` [-h] | [<-t target | -c config_file> [-v]]" >&2
}
```

2. Основи програмування мовою командного інтерпретатора bash

(продовження)

```
before_exit()
#Виводить прощальне повідомлення
{
   if [ $VERBOSE = on ]; then #Якщо включений говірливий режим
      if [ $COUNTER -eq 1 ]; then #Якщо $COUNTER дорівнює 1
         ENDING=""
         COPULA="is"
      else                  #Якщо $COUNTER не дорівнює 1 (більше 1, на ділі)
         ENDING="s"
         COPULA="are"
      fi
      echo "$COUNTER host$ENDING $COPULA probed"
      echo "Bye..."
   fi
}
```

(продовження)

```
while [ $# -gt 0 ]      #$# - кількість проініціалізованих позиційних параметрів
do #Доки $# більше нуля
   case $1 in           #$1 - 1-й позиційний параметр, $2 - 2-й и т. д.
   -h)
      usage
      exit 0
      ;;
   -t)
      TARGET=$2
      ;;
   -c)
      CONFIG_FILE=$2
      ;;
   -v)
      VERBOSE=on
      ;;
   esac
   shift            #Зсув позиційних параметрів (на 1 позицію ліворуч)
done
```

(продовження)

```
if [ $VERBOSE = on ]; then
   echo "It's `date +%k:%M:%S` now. Pingall v1.0 is starting...“
fi

if [ ! -z $TARGET ]; then
   pingone
   COUNTER=1
   before_exit
   exit 0
fi

if [ -z $CONFIG_FILE ]; then
   echo "Configuration file is not defined" >&2      #Файл не визначений
   exit 1
fi
if [ ! -f $CONFIG_FILE ]; then
   echo "Configuration file $CONFIG_FILE is not exist" >&2   #Файл не існує
   exit 1
fi
```

2. Основи програмування мовою командного інтерпретатора bash

(завершення)

```
#Створення тимчасового конфігураційного файлу
TMP_CONFIG_FILE=`mktemp /tmp/pingall.XXXXXX` || exit 1

#Зондування цілей, заданих у конфігураційному файлі
#grep видаляє порожні рядки та рядки коментарів (рядки, що починаються з #)
#tee зберігає цілі у тимчасовому файлі
cat $CONFIG_FILE | grep -E -v "^#|^$" | tee $TMP_CONFIG_FILE | while read TARGET
do
   pingone
done

#Підрахунок числа прозондованих вузлів
COUNTER=`grep -c ".*" $TMP_CONFIG_FILE`

#Видалення тимчасового конфігураційного файлу
rm -f $TMP_CONFIG_FILE

#Виведення прощального повідомлення
before_exit
```

```
#!/bin/sh
CONFIG_FILE=$HOME/pingall-hosts #Ім'я конфігураційного файлу
TARGET=                 #Ціль
VERBOSE=off             #Перемикач говірливості
COUNTER=0               #Лічильник прозондованих вузлів
pingone()
#Зондує один віддалений вузол
{
   ping -n -c3 -l3 -w2 $TARGET; echo
   #Інкремент лічильника прозондованих вузлів
   COUNTER=`expr $COUNTER + 1`
}

usage()
#Виводить інформацію про синтаксис
{
   echo "usage: `basename $0` [-h] | [<-t target | -c config_file> [-v]]" >&2
}
```

2. Основи програмування мовою командного інтерпретатора bash

(продовження)

```
before_exit()
{
    if [ $VERBOSE = on ]; then
        if [ $COUNTER -eq 1 ]; then
            ENDING=""
            COPULA="is"
        else
            ENDING="s"
            COPULA="are"
        fi
        echo "$COUNTER host$ENDING $COPULA probed"
        echo "Bye..."
    fi
}
```

2. Основи програмування мовою командного інтерпретатора bash

(продовження)

```
read_target()
{
    while read TARGET
    do
        if [ ! -z "$TARGET" -a `expr match "$TARGET" "#"` -eq 0 ]; then
            return 0
        fi
    done
    return 1
}
```

```
while getopts ht:c:v OPTION        #(продовження)
do
   case $OPTION in
   h)
       usage
       exit 0
       ;;
   t)
       TARGET=$OPTARG
       ;;
   c)
       CONFIG_FILE=$OPTARG
       ;;
   v)
       VERBOSE=on
       ;;
   \?)       #Невідома опція
       usage
       exit 1
   esac
done
```

2. Основи програмування мовою командного інтерпретатора bash

(продовження)

```
if [ $VERBOSE = on ]; then
    echo "It's `date +%k:%M:%S` now. Pingall v1.0 is starting..."
fi

if [ ! -z $TARGET ]; then
    pingone
    before_exit
    exit 0
fi

if [ -z $CONFIG_FILE ]; then
    echo "Configuration file is not defined" >&2          #Файл не заданий
    exit 1
fi
if [ ! -f $CONFIG_FILE ]; then
    echo "Configuration file $CONFIG_FILE is not exist" >&2   #Файл не існує
    exit 1
fi
```

(завершення)

```
#Підключення стандартного потоку введення до конфігураційного файлу
exec 3<&0 0<$CONFIG_FILE

#Зондування цілей, заданих у конфігураційному файлі
while read_target
do
   pingone
done

#Відновлення стану стандартного потоку введення
exec 0<&3

#Виведення прощального повідомлення
before_exit
```