

# Skripty a funkce

SKRIPTY	
<b>true</b>	vrátí úspěšný (nulový) návratový kód
<b>false</b>	vrátí neúspěšný (nenulový) návratový kód
<p><b>test</b> &lt;podminka&gt;   [ &lt;podminka&gt; ]   [[ &lt;podminka&gt; ]]</p>	<p>vyhodnotí danou podmínku a v závislosti na jejím splnění vrátí úspěšný či neúspěšný návratový kód; k testování existence specifických souborů slouží následující volby, jejichž argumentem je název souboru: <b>-e</b> jakýkoliv soubor, <b>-f</b> běžný soubor, <b>-b</b> blokové zařízení, <b>-c</b> znakové zařízení, <b>-d</b> adresář, <b>-h</b> symbolický odkaz, <b>-s</b> soubor s nenulovou velikostí, <b>-r</b> soubor s právem ke čtení, <b>-w</b> soubor s právem k zápisu, <b>-x</b> soubor s právem ke spuštění, <b>-u</b> soubor s SUID bitem, <b>-g</b> soubor s SGID bitem, <b>-k</b> soubor se sticky bitem, <b>-p</b> pojmenovaná roura, &lt;soubor1&gt; <b>-nt</b> &lt;soubor2&gt; soubor1 je novější než soubor2, &lt;soubor1&gt; <b>-ot</b> &lt;soubor2&gt; soubor1 je starší než soubor2; k testování celých čísel slouží operátory: <b>-eq</b> je rovno, <b>-ne</b> není rovno, <b>-lt</b> menší než, <b>-le</b> menší nebo rovno, <b>-gt</b> větší než, <b>-ge</b> větší nebo rovno; k testování řetězců: <b>=</b> je rovno, <b>!=</b> není rovno, <b>-z</b> &lt;retezec&gt; délka řetězce je nulová, <b>-n</b> &lt;retezec&gt; délka řetězce je nenulová; rozšířené testování umožňují operátory: <b>-a</b> logické A (oba výrazy jsou pravdivé), <b>-o</b> logické NEBO (alespoň jeden výraz je pravdivý), <b>!</b> logické NE (výraz je nepravdivý); při použití <b>[[ ]]</b> lze navíc testovat výrazy jako &lt;retezec&gt; <b>=</b> / <b>==</b> &lt;vzor&gt; (řetězec obsahuje vzor), &lt;retezec&gt; <b>!=</b> &lt;vzor&gt; (řetězec neobsahuje vzor), &lt;retezec&gt; <b>==</b> &lt;vzor&gt; (řetězec odpovídá regulárnímu výrazu), <b>!</b> &lt;retezec&gt; <b>==</b> &lt;vzor&gt; (řetězec neodpovídá regulárnímu výrazu), &lt;retezec1&gt; &lt; &lt;retezec2&gt; nebo &lt;retezec1&gt; &gt; &lt;retezec2&gt; (řetězec1 podle abecedy předchází nebo následuje řetězec2) a využít <b>&amp;&amp;</b> pro vyhodnocení druhého výrazu pokud první skončil úspěšně, <b>  </b> pro vyhodnocení druhého výrazu pokud první skončil neúspěšně a <b>()</b> pro seskupování výrazů</p> <pre>\$ test -f *.txt   \$ [ -f *.txt ] (návratový kód je 0, existuje-li právě 1 soubor s příponou „.txt“) \$ [[ -n \$(find . -name "*.txt") ]] (návratový kód je 0, existuje-li 1 a více souborů s příponou „.txt“) \$ [ ! -d PROD ] (návratový kód je 0, pokud adresář neexistuje) \$ [[ abc == *c ]] (návratový kód je 0, obsahuje-li řetězec „abc“ vzor „*c“) \$ [[ "abc" =~ .*c ]] (návratový kód je 0, odpovídá-li řetězec „abc“ regulárnímu výrazu „.*c“) \$ [[ a &lt; z ]] (návratový kód je 0, předchází-li řetězec „a“ v abecedním pořadí řetězec „z“) \$ [[ \$(awk -F ":" '/^root/{print \$4}' /etc/passwd) -eq 0 ]] (návratový kód je 0, je-li GID uživatele root 0) \$ [ \$? -ne 0 ] &amp;&amp; echo \$? (vypíše návratový kód předchozího příkazu, skončil-li neúspěšně) \$ [ \$PWD = \$HOME ] &amp;&amp; echo "doma"    pwd (vypíše „doma“, nachází-li se přihlášený uživatel v domovském adresáři, jinak zobrazí cestu k pracovnímu adresáři) # duid=\$(awk -F ":" '{print \$3}' /etc/passwd   sort   uniq -d); [[ -z "\$duid" ]] &amp;&amp; echo "NONE"    echo "\$duid" (vypíše duplicitní UID v systému)</pre>
<p><b>if</b> &lt;seznam&gt;; <b>then</b> &lt;seznam&gt;; [<b>elif</b> &lt;seznam&gt;; <b>then</b> &lt;seznam&gt;;] ... [<b>else</b> &lt;seznam&gt;;] <b>fi</b></p>	<p><b>if</b> uvádí podmínku, je-li pravdou, pak <b>then</b> určí požadovanou akci, není-li pravdou, <b>elif</b> stanoví další podmínku a <b>then</b> určí další akci, jinak <b>else</b> určí akci alternativní, seznam podmínek ukončuje <b>fi</b></p> <pre>#!/bin/bash # Skript kontroluje, zda je přihlášený uživatel root. Pokud je uživatel root, vyzve k zadání příkazů. Pokud uživatel není root, informuje jej, že nemá povoleno spouštět skript a ukončí se s chybovou zprávou.  if [[ \$EUID -ne 0 ]]; then     echo "Uživatel '\$(whoami)' není oprávněn spustit \${0##*/}."     exit 1 fi read -p "Zadejte své příkazy: " prikazy eval "\$prikazy" echo "Skript byl úspěšně dokončen."</pre> <p><b>#!/bin/bash</b>  <b># Skript ověří, zda je zadaný balíček nainstalován.</b></p> <pre>echo -n "Zadejte název balíčku: " while read PACKAGE do     rpm -q \$PACKAGE &gt; /dev/null     if [ \$? -eq 0 ]     then         echo "ANO, \$PACKAGE je nainstalován."     else         echo "NE, \$PACKAGE není nainstalován."     fi done . \$0 done</pre>

SKRIPTY	
<p><b>case &lt; slovo&gt; in &lt; vzor&gt;[&lt; vzor&gt;...] &lt; seznam&gt;; esac</b></p>	<p>vykoná příkaz na základě shody slova se vzorem; slovo představuje libovolnou proměnnou, vzor „*“ uvádí akci, pokud shoda nenastane, seznam vzorů ukončuje „)“ a skupina příkazů se zakončí „;“</p> <pre>#!/bin/bash # Skript podle operace nainstaluje, aktualizuje či odinstaluje daný balíček.  echo -n 'Zadejte operaci a balíček: ' read OPERACE BALICEK case \$OPERACE in   nainstaluj)    yum install -y \$BALICEK;;   aktualizuj)   yum update -y \$BALICEK;;   odinstaluj smaz) yum remove -y \$BALICEK;;   *)            echo 'Neznama operace.'                 echo 'Pouziti: {nainstaluj aktualizuj odinstaluj smaz} &lt;balicek&gt;;'; esac</pre>
<p><b>read &lt; promenna ...&gt;</b></p>	<p>čte řádek ze STDIN, rozdělí ho na slova a ta přiřadí danému počtu proměnných v odpovídajícím pořadí pro další zpracování; pokud je více slov než proměnných, zbývající slova a jejich oddělovače jsou přiřazeny poslední proměnné, pokud je proměnných více než slov, zbývajícím proměnným jsou přiřazeny prázdné hodnoty, <b>-a &lt; indexované_pole&gt;</b> určí indexované pole, kde jsou slova přiřazena k sekvenčním indexům, <b>-n &lt;n&gt;</b> čte pouze daný počet znaků, <b>-p &lt;vyzva&gt;</b> určí výzvu, která se uživateli zobrazí před čtením vstupu, <b>-r</b> ruší znaku „\“ speciální význam, <b>-s</b> nezobrazí žádné znaky na vstupu, <b>-t &lt;n&gt;</b> nastaví časový interval v sekundách pro čtení vstupu</p> <pre>\$ cat cisla.txt   { read a; read b; read c; echo "\$a/\$b/\$c";} (čte první tři řádky souboru, které vypíše na jeden řádek za sebou oddělené lomítkem)  #!/bin/bash # Skript vyzve uživatele k zadání jména a hesla a potvrdí jeho přijetí.  echo -n "Jmeno: "; read jmeno echo -n "Heslo: "; read -s heslo echo echo "Heslo pro \$jmeno zadano."</pre>
<p><b>while &lt; seznam1&gt;; do &lt; seznam2&gt;; done</b></p>	<p>spouští opakovaně „seznam2“, dokud „seznam1“ vrácí úspěšný návratový kód</p> <pre>#!/bin/bash # Skript otvírá čtyři terminálová okna.  i=0 while [ \$i -lt 4 ]; do xterm &amp;   i=\$((i+1)); done  #!/bin/bash # Skript vytváří soubor s informacemi o právě přihlášených uživateli v intervalu pěti minut.  while true; do   who &gt; info-\$(date   awk '{print \$2,\$3,\$4,\$6}'   sed 's/ /_/g').txt   sleep 300; done  #!/bin/bash # Skript vyzve uživatele k zadání jména vzdáleného počítače, na kterém spustí příkaz, který zjistí verzi distribuce Linuxu a výsledek uloží do souboru na lokálním počítači.  echo -n "Jmeno pocitace: "; while read hostname; do ssh \$hostname 'echo "Distribuce Linuxu": \$(cat /etc/redhat-release)' &gt; \${hostname}_info &amp;&amp; echo -n "Jmeno pocitace: "; done</pre>

SKRIPTY	
<pre>until &lt;seznam1&gt;; do &lt;seznam2&gt;; done</pre>	<pre>spouští opakovaně „seznam2“, dokud „seznam1“ vrací neúspěšný návratový kód #!/bin/bash # Skript vypíše čísla od 0 do 99.  a=-1 until [ \$a -eq 99 ]; do     a=\$(expr \$a + 1)     echo \$a; done  #!/bin/bash # Skript kopíruje soubory z domovského do webového adresáře, kde každou # hodinu vytvoří nový adresář, přičemž v případě jeho zaplnění na více než 90 % # smaže adresáře starší 30 dní.  while true; do     DISKFUL=\$(df -h \$WEBDIR   grep "^/"   \     awk '{print \$5}'   cut -d "%" -f1 -)      until [ \$DISKFUL -ge 90 ]; do         WEBDIR=/var/www/webcam         DATE=\$(date +%Y%m%d)         HOUR=\$(date +%H)         mkdir \$WEBDIR/"\$DATE"          while [ \$HOUR -ne 00 ]; do             PICDIR=/home/user/pics             DESTDIR=\$WEBDIR/"\$DATE"/"\$HOUR"             mkdir "\$DESTDIR"             cp \$PICDIR/*.jpg "\$DESTDIR"/             sleep 3600             HOUR=\$(date +%H); done          DISKFULL=\$(df -h \$WEBDIR   grep "^/"   \         awk '{print \$5}'   cut -d "%" -f1 -)         done          TOREMOVE=\$(find \$WEBDIR -type d -mtime +30)         for i in \$TOREMOVE; do             rm -rf "\$i"; done         done</pre>

SKRIPTY	
<pre>for &lt;promenna&gt; [in &lt;slovo&gt; ...] do &lt;seznam&gt;; done</pre>	<pre>přihadí proměnné postupně hodnotu všech slov nebo všech pozičních parametrů skriptu a provede seznam příkazů pro každou z položek #!/bin/bash # Skript zobrazí každý argument příkazového řádku spolu s jeho odpovídajícím číslem.  count=1  for arg; do     echo "Argument \$count: \$arg"     ((count++)) done  #!/bin/bash # Skript vytvoří uživatele uvedené v souboru „uzivatele“ v pracovním adresáři, přidá je do skupiny „admins“ a nastaví heslo „password“.  for uzivatel in \$(cat uzivatele); do     useradd -g admins \$uzivatel     echo password   passwd --stdin \$uzivatel done  #!/bin/bash # Skript vypíše obsah pracovního adresáře.  for x in \$(ls -F) do     echo "V adresari \$(pwd) se nachazi soubor ci adresar: \$x" done  #!/bin/bash # Skript spočítá počet položek v pracovním adresáři.  POC=0 for nazev in * do     POC=\$((POC+1)) done echo "V adresari \$(pwd) je \$POC polozek."  #!/bin/bash # Skript změní formát všech souborů .tif na .jpg.  for pic in *.tif; do     convert "\$pic" "\$(echo "\$pic"   sed 's/\.tif/.jpg/')" done  #!/bin/bash # Skript zjistí, které z uvedených procesů běží, vypíše jejich jména včetně počtu řádků, které zabírají a nakonec zobrazí výpis všech běžících procesů a celkový počet řádků.  services="httpd pmon lsnr sap" for service in \$services do     running_processes=\$(ps -ef   grep \$service   egrep -v 'grep vnc client API')     if [ -n "\$running_processes" ]     then         echo \$service:         echo \$running_processes         echo "radku: \$(echo "\$running_processes"   wc -l)"         echo "======"     fi done echo echo "Souhrn:" echo ps -ef echo echo "radku: \$(ps -ef   wc -l)"  #!/bin/bash # Skript se připojí ke všem serverům, jejichž IP adresy jsou uvedeny v souboru „servers“, na nich spustí příkazy, které zjistí jméno serveru a všechny jeho IP adresy a výsledek zapíše do souboru v lokálním počítači.  for ip in \$(&lt; servers); do ssh \$ip 'hostname; ifconfig   awk '/inet / {print \$2}'   grep -v 127.0.0.1' &gt;&gt; servers_info.txt; done  #!/bin/bash # Skript se připojí ke vzdáleným počítačům, na kterých nainstaluje software pro danou službu, kterou poté nainstaluje a ověří její status.  for x in {a..c}; do echo "=== node\${x} ==="; ssh node\${x} 'yum install -y device-mapper-multipath; systemctl enable --now multipathd; systemctl status multipathd'; echo; done</pre>

## SKRIPTY

přihadí proměnné postupně hodnotu všech slov nebo všech pozičních parametrů skriptu, jejichž očíslovaný seznam zobrazí společně s výzvou pro uživatele k zadání čísla položky a poté provede seznam příkazů pro vybranou položku (interaktivní obdoba příkazu „for“); přečtený řádek se uloží do proměnné „REPLY“

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

# Skript zobrazí menu s nabídkou služeb. Po vybrání jedné z nich se objeví vnořené menu s nabídkou operací, které lze provést.

```
while true; do
PS3="Vyberte službu nebo odejdete: "
select svc in cups httpd sshd odejit
do
  case $svc in
    cups)
      PS3="Vyberte operaci nebo odejdete: "
      select opt in start stop restart status odejit
      do
        case $opt in
          start)
            systemctl start cups
            break;;
          stop)
            systemctl stop cups
            break;;
          restart)
            systemctl restart cups
            break;;
          status)
            systemctl status cups
            break;;
          odejit)
            break;;
          *)
            echo "Neplatná volba $REPLY"
            break;;
        esac
      done
      break;;
    httpd)
      PS3="Vyberte operaci nebo odejdete: "
      select opt in start stop restart status odejit
      do
        case $opt in
          start)
            systemctl start httpd
            break;;
          stop)
            systemctl stop httpd
            break;;
          restart)
            systemctl restart httpd
            break;;
          status)
            systemctl status httpd
            break;;
          odejit)
            break;;
          *)
            echo "Neplatná volba $REPLY"
            break;;
        esac
      done
      break;;
    sshd)
      PS3="Vyberte operaci nebo odejdete: "
      select opt in start stop restart status odejit
      do
        case $opt in
          start)
            systemctl start sshd
            break;;
          stop)
            systemctl stop sshd
            break;;
          restart)
            systemctl restart sshd
            break;;
          status)
            systemctl status sshd
            break;;
          odejit)
            break;;
          *)
            echo "Neplatná volba $REPLY"
            break;;
        esac
      done
      break;;
    odejit)
      break 2;;
  *)
    echo "Neplatná volba $REPLY";;
  esac
done; done
```

**select <promenna> [in <slovo> ...] do <seznam>; done**

SKRIPTY	
<b>break</b> [<n>]	<p>ukončí definitivně daný cyklus ve smyčce „for“, „while“, „until“, nebo „select“ na základě určitých kritérií, n určí počet úrovní vnořených cyklů</p> <pre>#!/bin/bash # Skript spustí smyčku, která by měla procházet cykly od 1 do 5, ale předčasně ji ukončí během třetího cyklu, když hodnota proměnné „cyklus“ dosáhne 3. for cyklus in {1..5}; do   if [ "\$cyklus" -eq 3 ]; then     echo "Prerušuji během cyklu \$cyklus"     break   fi   echo "Cyklus: \$cyklus" done echo "Smyčka skončila."</pre>
<b>continue</b> [<n>]	<p>přeskočí daný cyklus ve smyčce „for“, „while“, „until“, nebo „select“ na základě určitých kritérií a pokračuje v dalším, n určí úroveň cyklu</p> <pre>#!/bin/bash # Skript spustí smyčku, která by měla procházet cykly od 1 do 5, ale přeskočí třetí cyklus, když hodnota proměnné „cyklus“ dosáhne 3 a pokračuje v dalším cyklu. for cyklus in {1..5}; do   if [ "\$cyklus" -eq 3 ]; then     echo "Preskakují cyklus \$cyklus"     continue   fi   echo "Cyklus: \$cyklus" done  #!/bin/bash # Skript převádí velká písmena v názvech souborů na malá. LIST=\$(ls) for nazev in \$LIST; do   if [[ \$nazev != *[:upper:]* ]]; then     continue; fi   ORIG=\$nazev   NEW=\$(echo \$nazev   tr 'A-Z' 'a-z')   mv \$ORIG \$NEW   echo "Nový název pro \$ORIG je \$NEW" done</pre>
<b>getopts</b> <seznam_voleb> <jmeno> [<argumenty>]	<p>rozdělí (zpracuje) poziční parametry (volby a argumenty), které jsou předány shellovému skriptu; &lt;seznam_voleb&gt; určí seznam podporovaných znaků voleb, následuje-li za znakem dvojtečka (:), očekává se, že volba bude mít argument, je-li dvojtečka před prvním znakem, potlačí se hlášení chyb; při každém volání „getopts“ získá další volbu z pozičních parametrů a umístí znak volby do proměnné &lt;jmeno&gt;, pokud volba neodpovídá těm, které jsou definovány v &lt;seznam_voleb&gt;, „getopts“ nastaví proměnnou &lt;jmeno&gt; na otazník (?) a pokud je potlačeno hlášení chyb, nalezený znak volby se umístí do proměnné „OPTARG“; vyžaduje-li volba argument, „getopts“ umístí tento argument do proměnné „OPTARG“, pokud není nalezen požadovaný argument a je potlačeno hlášení chyb, umístí se dvojtečka do proměnné &lt;jmeno&gt; a „OPTARG“ se nastaví na nalezený znak volby; při spuštění skriptu musí volby, které vyžadují argument, následovat jejich argumenty, než se použije další volba</p> <pre>#!/bin/bash # Skript povolí pro své spuštění používat pouze přesně definované volby a očekávané argumenty.  pouziti() {   echo "Pouziti: \$0 {-x -y &lt;retezec&gt;}"   exit 1 }  [[ ! "\$@" =~ -.+ ]] &amp;&amp; pouziti  while getopts :xy: volba do   case \$volba in     x)       echo "Je použita volba '-x'."       ;;     y)       echo "Je použita volba '-y' s argumentem '\$OPTARG'."       ;;     \?)       echo "Nepřijatelná volba '-\$OPTARG'."       pouziti       ;;     :)       echo "Volba '-\$OPTARG' vyžaduje argument."       pouziti       ;;   esac done</pre>
<b>exit</b> [<n>]	<p>ukončí skript s daným návratovým kódem, není-li uveden, návratovým kódem skriptu je návratový kód posledního příkazu spuštěného uvnitř skriptu</p>

FUNKCE	
<pre>function &lt;funkce&gt; { &lt;seznam&gt;; }   &lt;funkce&gt; () { &lt;seznam&gt;; }</pre>	<p>definuje funkci, jejíž obsah je uzavřen do složených závorek; funkce se provádí v běžném shellu a využívá se zejména v případech, kdy je třeba provést danou operaci (posloupnost příkazů) opakovaně; spustí se zadáním jejího názvu a standardně vrací návratový kód posledního provedeného příkazu, není-li stanoveno jinak příkazem „return“; uvnitř funkce lze příkazem „local“ definovat lokální proměnnou (ta se dědí i do vnořených funkcí), pokud existuje globální proměnná stejného názvu, je ve funkci potlačena; funkci lze používat trvale (i po dalším spuštění shellu) jejím uvedením v <i>~/.bashrc</i>; informace o argumentech předaných skriptům či funkcím jsou uloženy ve zvláštních proměnných („<i> \$# </i>“ = celkový počet argumentů, „<i> \$&lt;n&gt; </i>“ = daný argument v pořadí v rozmezí 1-9, „<i> \${&lt;n&gt; } </i>“ = daný argument v pořadí v libovolném rozmezí, „<i> \$* </i>“ či „<i> @\$ </i>“ = všechny argumenty)</p> <p># Funkce „test“ vytvoří soubor, který je zadán jako argument a nastaví mu příslušná přístupová práva. Pokud soubor stejného jména již existuje nebo zadaný název obsahuje pouze mezeru či zcela chybí, vypíše se chybové hlášení.</p> <pre>function test { if ! [[ -e \$*    \$* = ' ' ]]; then     touch \$*; chmod 755 \$* else     echo "Chyba" fi echo "Počet argumentů: \$#"</pre> <p>echo "Všechny argumenty: \$*" echo "Druhý argument: \$2" } <p># Funkce „uzivatel“ zjistí, zda uživatel, který je zadán jako argument, existuje v systému, a pokud ano, vypíše o něm podrobné informace.</p> <pre>uzivatel () { if [ \$(grep -ic ^\$1 /etc/passwd) -eq 0 ]; then     echo "Uzivatel \$1 neexistuje." else     echo "Uzivatel \$1 existuje." &amp;&amp; finger \$1 fi }</pre> <p># Funkce „procento“ vypočítá procentuální hodnotu druhého číselného argumentu z prvního, přičemž první argument představuje hodnotu 100 %.</p> <pre>procento () { local x=\$1 y=\$2 printf "\$x = 100 %%\n\$y = x %%\n" echo "\$y = \$[( ( 100 * \$y ) ) / \$x] % z \$x" }</pre> <p># Funkce „pouziti“ vypíše možné volby pro spuštění skriptu, pokud nebyla žádná platná volba použita, vrátí návratový kód 1.</p> <pre>pouziti () { echo "Pouziti: \$0 {-start -stop -restart}"; exit 1; }</pre> </p>

FUNKCE	
<b>local</b> [<promenna>[=<hodnota>] ...]	definuje lokální proměnnou funkce, <b>-a</b> určí indexované pole, <b>-A</b> určí asociativní pole; bez argumentu zobrazí všechny lokální proměnné funkce
<b>return</b> [<n>]	ukončí funkci s daným návratovým kódem, není-li uveden, návratovým kódem funkce je návratový kód posledního příkazu spuštěného uvnitř funkce

From:

<https://prompt.cz/> - **Prompt.cz**

Permanent link:

<https://prompt.cz/skripty-a-funkce>

Last update: **2024/05/17 17:22**

