Lexikální funkce

Nápověda: help lexicals, help lex

Slouží ke zjištění informací o systému, procesech, souborech a adresářích, pro práci s řetězy znaků. Jména jsou ve tvaru **f$funkce()**. U bezparametrických funkcí nutno psát prázdné závorky.

Při použití mimo výraz se píše `'f$funkce()`, uvnitř textového řetězce pak `''f$funkce()`. Tedy analogie se symboly.

```
a = f$time()
    show symbol a

f$directory()
    aktuální adresář

f$mode()
    INTERACTIVE, BATCH, NETWORK

f$verify()
    0,1 režim verifikace (trasování)

f$length("Ahoj")
    4

f$extract(0,2,"test")
    "te"

f$locate("bry","Dobry den")
    2    nebo f$length("Dobry den") při nenalezení

f$element(0," ","Dobre rano")
    "Dobre" (nebo oddělovač, 2. parametr)
    pořadí elementu, oddělovač, řetěz znaků

f$element(1,"r","Dobre rano")
    "e "

f$edit(string,operation)
    compress, trim, upcase, lowercase

f$parse("[adr]fil.typ",,,"name")
          "name"          "fil"
          "node"          "cs::"
          "device"        "student$device:"
          "directory"     "[adr]"
          "type"          ".typ"
          "version"       ";1"
    Analýza zadané specifikace souboru

f$search("[...]*.*")      "[a]prvni.soubor"
                          "[b]druhy.soubor"
                          ""
Opakovaným voláním se zjišťují všechny soubory
odpovídající zadané masce. Po jejich vyčerpání
je výsledkem "".

f$type(expression)       "string", "integer", ""
    Typ výrazu nebo "" při nedefinovaném symbolu

f$fao        formátovaný ASCII výstup, jako printf
    n =14
    s = f$fao("Vysledek !5ZL !OW !XW",n,n,n)
       Vysledek 00014 000016 000E

f$file_attributes(filespec, item)
    Dotaz na vlastnosti souboru
    velikost, datum vytvoření, ....

f$device
    Zjisti jména všech zařízení daného typu,
    analogie s f$search.
    Např. f$device(,"disk") zjišťuje disky.

f$getdvi(device, item)
     Dotaz na vlastnosti nějakého zařízení,
     např. volné místo na disku.
```

Příkazová procedura

- textový soubor, implicitně typu .com
- `@soubor`
  spustí příkazovou proceduru v rámci aktuálního procesu
- `submit soubor/after=...`
  zařadí do fronty dávkových úloh, spustí v zadaný čas
  (místní aliasy pro práci s frontou jsou `qli a qdl`)
- na začátku příkazového řádku se píše `$`
  (odlišení příkazů a dat)
- pokračovací řádek, předchozí končí pomlčkou
- za `!` následuje komentář
- trasování
  `set verify, set noverify`

```
$ toto je prilis dlouhy prikaz, ktery -
se nevejde na jednu radku ! komentar
$
$! A ted to prijde
$ dir
```

Logická jména

- jiné označení souboru, adresáře, zařízení
- náhrada dlouhých specifikací
- přenositelnost, nezávislost na konkrétním zařízení či adresáři
- lze použít všude tam, kde se užívá specifikace souboru
- použití na úrovni DCL i aplikačního programu
- interpretace na úrovni syst. služeb (open, create file)

Příklad definice a užití logického jména

```
$ define data user$device:[student.xnovak]vstup.txt
$ type data
$ show logical data
$ deassign data    ! a na zaver jmeno zrusit


$! Logické jméno ukazující na adresář
$ define sys$login user$device:[student.xnovak]
$ dir sys$login
$ type sys$login:login.com
```

Hierarchie tabulek logických jmen

proces	`define`	implicitní
job, úloha	`define/job`	proces a jeho podprocesy
skupina	`define/group`	procesy skupiny uzivatelu (gid)
systém	`define/system`	všechny procesy

Prohledává se počínaje tabulkou procesu, platí první výskyt.

Poznámka: Lze vytvořit i jiné tabulky a včlenit je do hierarchie, lze stanovit přístupová práva a ACL k tabulce.

Režimy přístupu k log. jménu

user mode	`define/user`	platnost jen běh příštího programu
supervisor mode	`define`	implicitní, platnost do zániku procesu
executive mode	`define/exe`	využíváno privilegovanými programy

Privilegované programy mohou např. vyhledávat jen logická jména s režimem přístupu executive mode.

Standardní vstup a výstup

Soubory představující std. vstup a výstup jsou dány logickými jmény. Oproti ostatním jménům jde o tzv. process permanent logical names, tedy vlastně nezrušitelná logická jména.

`sys$input`	standardní vstup
`sys$output`	stadardní výstup
`sys$error`	soubor pro výstup chybových zpráv
`sys$command`	počáteční nastavení `sys$input` v okamžiku vzniku procesu

Nastavení std. vstupu a výstupu

	interakt. práce	interaktivní příkazová procedura	dávková úloha
sys$command	terminál	terminál	soubor
sys$input	terminál	soubor	soubor
sys$output	terminál	terminál	.log soubor
sys$error	terminál	terminál	.log soubor

Přesměrování výstupu

```
$  define sys$output out.dat
$  dir
$  type out.dat       ! pozor
$  deassign sys$output
$  type out.dat

$! Jinak, omezena platnost log. jmena
$  define/user sys$output out.dat
$  dir
$  type out.dat

$! prikaz directory umi ve skutecnosti i toto:
$  dir/output=out.dat
```

Přesměrování vstupu

```
$! vytvor soubor, vloz data ze std. vstupu
$  create data.dat
toto jsou dva radky
meho datoveho souboru
$  type data.dat
```

A poměrně oblíbená konstrukce:

```
$! cti sys$input, opisuj na sys$output
$  type  sys$input
Napoveda k programu
1 - zadani parametru
...
9  - konec
$  inquire  volba  "Tvoje volba"
```

Předefinování jmen souborů

Program `program` otevírá soubory `in-data` a `out-data`. Skutečně použité soubory jsou `vstup.dat` a `výstup.dat`.

```
$ define/user in-data  vstup.dat
$ define/user out-data vystup.dat
$ run  program
$! Konec platnosti presmerovani
```

V příkazové proceduře se aktivuje program, který čte data z klávesnice. Pro jeho správnou funkci je třeba přesměrovat vstup zpět na původní nastavení, na terminál.

```
$ define/user sys$input  sys$command
$ edit vystup.dat
```

Zpracování souboru

Soubor je třeba otevřít, postupně zpracovat a uzavřít. Při otevření se definuje logické jméno, kterým se na otevřený soubor odkazujeme (file handle).

```
$ open/read log_name data.txt
$ read log_name line
      ...
$ read log_name line
$ close log_name
```

Způsob otevření souboru: pro čtení, zápis (nový soubor, nová verze) nebo otevření existujícího souboru na jeho konci pro zápis.

```
$ open/read       log_name  data.txt
$ open/write      log_name  data.txt
$ open/append     log_name  data.txt
```

Čti soubor po jednotlivých záznamech (řádcích) a opisuj je na std. výstup.

```
$  open/read soubor 'p1'
$! v symbolu p1 je 1. parametr z prikazoveho radku
$cyklus:
$! cti zaznam do symbolu radek
$! pri vycerpani souboru prejdi na navesti konec
$  read/end_of_file=konec soubor radek
$  write sys$output radek
$  goto cyklus
$konec:
$  close soubor
```

Podmíněný příkaz

Podmíněný příkaz má dvě podoby. Zkrácenou, kdy lze provést jediný příkaz. V plné podobě exituje i s větví `else` a možností zápisu více příkazů v každé větvi.

```
$ if  expression  then  command

$ if expression
$ then   [command]
$         command
          . . .
$        command
$ else                        ! nepovinná část
$         command
$ endif                          ! povinné zakončení
```

numerické operátory: `.eq. .ne. .gt. .ge. .lt. .le.`
řetězcové operátory: `.eqs. .nes. .gts. .ges. .lts. .les.`
logické operátory: `.and. .or. .not.`

Při operaci .not. se invertují všechny bity.

```
$ if radek .eqs. "" then goto konec
$ if ano then goto pokracuj
```

[Příklad - počáteční písmena]

```
$! Kopirovani souboru
$! Jmena soubopru mohla byt na prikazovem radku
$! Pokud nebyla, tak se na ne zeptam
$
$ if "''p1'" .nes. "" then goto open_infile
$ inquire p1 "Vstupni soubor"
$open_infile:
$ open/read/error=openerr infile 'p1'
$
$ if "''p2'" .nes. "" then goto open_outfile
$ inquire p2 "Vystupni soubor"
$open_outfile:
$ open/write outfile 'p2'
$
$loop:
$ read/end_of_file=loop_end infile line
$ outline :=
$ i = 0
$lineloop:
$ word = f$element(i," ",line)
$ i = i+1
$ if "''word'" .eqs. " " then goto writeout
$ outline = "''outline'''f$extract(0,1,word)"
$ goto lineloop
$writeout:
$ write outfile outline
$ goto loop
$
$loop_end:
$ close infile
$ close outfile
```

```
$ exit
$
$openerr:
$ write sys$output "Vstupni soubor se nepovedlo otevrit"
```

Příklad - informace o souborech

```
$! Prace se skupinou souboru
$! Urci jmeno a vlastnosti souboru
$
$ say := write sys$output
$ if "''p1'" .nes. "" then goto main_loop
$ inquire p1 "Specifikace skupiny souboru"
$
$main_loop:
$ fil = f$search(p1)
$ if "''fil'" .eqs. "" then exit
$ say "Plna specifikace: ''fil'"
$ fname = f$parse(fil,,,"NAME")
$ ftype = f$parse(fil,,,"TYPE")
$ forg = f$file_attributes(fil,"ORG")
$ say "Jmeno, organizace ''fname'''ftype' ''forg'"
$
$! Totez trochu jinak ...
$ say "Jmeno, typ a organizace: " -
  + f$parse(fil,,,"NAME") + f$parse(fil,,,"TYPE") -
  + "  ''f$file_attributes(fil,""ORG"")"
$ goto main_loop
```

Zadání úloh z operačních systémů (OSY/VMS) - letní semestr 2003/2004

Zadané úlohy vypracujte v příkazovém jazyce DCL. Orientujte se na využívání lexikálních služeb. Vyhýbejte se používat unixové triky, nadbytečné přesměrování vstupů a výstupů, neužívejte programy přenesené z unixového prostředí (pipe, sed awk, grep).

Pozitivně se hodnotí použití VMS-ového přístupu, elegance řešení, efektivita, nápad.

1. V aktuálním adresáři nalezněte všechny soubory, jejichž typ obsahuje pouze souhlásky. Vypište i jejich celkový počet.
2. V aktuálním adresáři nalezněte všechny soubory, které byly vytvořeny v lichou vteřinu. Vypište i jejich celkový počet.
3. Zobrazte názvy všech podadresářů, které obsahují lichý počet souborů.
4. Sčítačka dvou celých kladných čísel pracující s čísly až do velikosti 10^79.
5. Ze zadaného vstupního souboru vytvořte výstupní, který bude obsahovat v jednotlivých řádcích jen neopakující se souhlásky. Příklad:

```
6.   Vstup:   dobry den, jak se dnes mame?
     Vystup:  dbrnjksm
7.            toto je priklad
     tjprkld
```

8. Sčítačka/odčítačka pracující s celými čísly.
   Příklad vstupu: 15-3-2+8
9. Kolik souborů v aktuálním adresáři má jméno obsahující lichý počet písmen? (jen písmena od A do Z, ne všechny znaky).
10. Zkontrolujete, zda v zadaném textovém souboru jsou obsažena slova, která obsahují dvě po sobě jdoucí samohlásky. Tato slova zobrazte.
11. Zpracujte vstupní textový soubor a vytvořte výstupní. Ve výstupním souboru jsou vypuštěny nadbytečné mezery. Slova v řádcích výstupního souboru jsou v opačném pořadí, malými písmeny. Viz ukázka:

```
12.  Vstup:   Toto    je maly  priklad
13.  Vystup:  priklad maly je toto
```

14. Zpracujte vstupní textový soubor a vytvořte výstupní. Ve výstupním souboru jsou vypuštěny nadbytečné mezery. Jednotlivá slova jsou psána pozpátku dle vzoru:

```
15.  Vstup:   Dobry        den
16.  Vystup:  yrboD ned
```

17. Zpracujte vstupní textový soubor a vytvořte výstupní. Ve výstupním souboru jsou vypuštěny nadbytečné mezery. Samohlásky budou zapsány velkými písmeny.
18. Zpracujte vstupní textový soubor a vytvořte výstupní. Ve výstupním souboru jsou vypuštěny nadbytečné mezery. Řádky jsou po jednotlivých slovech přeformátovány tak, aby nebyly delší než zvolený limit, a přitom se mu co nejvíce blížily.
19. Zpracujte vstupní textový soubor a vytvořte výstupní. Ve výstupním souboru jsou vypuštěny nadbytečné mezery. Dále zjistěte celkový počet řádek, slov a písmen. Určete, která číslice má nejvyšší výskyt.
20. Zobrazte jméno a typ například pěti (obecně X) souborů s nejvyššími čísly verzí.
21. Zpracujte vstupní textový soubor a vytvořte výstupní. Ve výstupním souboru jsou vypuštěny nadbytečné mezery mezi slovy. Počáteční a koncový znak každého slova výstupního souboru bude zapsán velkým písmenem, ostatní malými písmeny.
22. Zpracujte vstupní textový soubor a vytvořte výstupní. Ve výstupním souboru jsou vypuštěny nadbytečné mezery. Ve výstupním souboru je opačné pořadí řádek, viz ukázka. Můžete použít pomocný pracovní soubor a příkaz SORT.

```
23.  Vstup:  prvni radek            Vystup:
     posledni
24.          druhy    radek                 druhy
     radek
25.          posledni                       prvni
     radek
```

26. Zpracujte vstupní textový soubor a vytvořte výstupní. Ve výstupním souboru jsou vypuštěny nadbytečné mezery a všechna slova uvedená v dalším, definičním souboru.
27. Zpracujte vstupní textový soubor a vytvořte výstupní. Slova výstupního souboru jsou sestupně uspořádána podle své délky. Můžete použít pomocný soubor a příkaz SORT.
28. Zobrazte jméno, typ a velikost například pěti (obecně X) největších souborů z aktuálního adresáře.
29. Zobrazte jméno, typ a velikost například pěti (obecně X) nejstarších soborů z aktuálního adresáře.
30. Ověřte, zda jednotlivé řádky textového souboru obsahují nejvýše N různých znaků (písmen).
31. Zobrazte jméno, typ, datum vzniku a počet řádků v souboru pro všechny soubory v aktuálním adresáři. Jména měsíců označujte českými zkratkami.
32. Zobrazte celkový počet bloků alokovaných soubory v aktuálním adresáři, skutečně (užitečně) využitou velikost souboru v blocích a jejich rozdíl. Výstup hezky formátujte.
33. Zjistěte jména všech diskových zařízení, počet volných bloků a maximální možný počet souborů na každém z nich. Výstup hezky formátujte.
34. Napište hodiny tikající zhruba každých pět sekund. Při každém tiknutí se zobrazí datum a čas v obvyklém systémovém tvaru, ale s českou zkratkou jména měsíce: 11-DUB-1999 14:30:16.18. Můžete používat příkaz WAIT.
35. Sledujte, za jak dlouho se na některém z diskových zařízení změní počet volných bloků o více, než je předem daný limit, např. 10 bloků. Uveďte dobu čekání na tuto událost a jméno disku.
36. Soubor ve svých řádcích obsahuje slova oddělovaná mezerami. Zjistěte počet slov, která jsou zápisem čísla. Určete jejich počet a také počet výskytu jednotlivých číslic.
37. Zobrazte jméno, typ a velikost například pěti (obecně X) souborů s nejdelším jménem.
38. Určete, zda textový soubor obsahuje jen vámi zvolená slova.
39. Zpracujte vstupní textový soubor a vytvořte výstupní. Slova výstupního souboru jsou vzestupně uspořádána podle své délky. Můžete použít pomocný soubor a příkaz SORT.
40. V aktuálním adresáři nalezněte všechny soubory, které nemají záložní verzi.