PB173 Perl

06 Súborový systém

Roman Lacko xlacko1@fi.muni.cz

2023-03-25

Obsah

Súborový systém

Súborový systém: Vytvorenie súboru

Perl nemá funkciu `creat(2)`.

```
open HANDLE, MODE, FILENAME
```

Režimy `>`, `+>`, `>>`, `+>>` vytvoria zadaný súbor.

```
sysopen HANDLE, FILENAME, MODE, [PERMS]
```

Systémovo závislé otvorenie súboru, v podstate `open(2)`.

Existujú aj funkcie `sysread`, `syswrite` a `sysseek`.

Súborový systém: Vytvorenie súboru

```
umask [EXPR]
```

Nastaví (a bez argumentu vráti) aktuálnu masku práv pre vytvorené súbory.

Súborový systém: Premenovanie

```
rename OLDNAME, NEWNAME
```

Zmení meno súboru ‹OLDNAME› na ‹NEWNAME›.

Ak ‹NEWNAME› existuje, prepíše sa.

Súborový systém: Odkazy

```
link OLDFILE, NEWFILE
symlink OLDFILE, NEWFILE
```

Vytvorí pevný resp. symbolický odkaz na súbor.

```
readlink [EXPR]
```

Prečíta a vráti cieľ symbolického odkazu.

```
unlink LIST
```

Odstráni zadané odkazy.

Po zrušení posledného odkazu na súbor kernel tento súbor zruší.

Súborový systém: Modul ‹File::Copy›

```
use File::Copy;

copy SOURCE, DESTINATION, [BUFFER_SIZE]
move SOURCE, DESTINATION
```

- ‹SOURCE› môže byť aj otvorený súbor
- ‹DESTINATION› môže byť aj adresár, vtedy ‹SOURCE› musí byť názov súboru

Súborový systém: Vlastnosti

```
stat [EXPR]
lstat
```

Vráti pole atribútov zo štruktúry ‹struct stat› systémového volania ‹stat(3)›. Argumentom môže byť názov súboru alebo File Handle.

```
my @passwd = stat "/etc/passwd";
say $passwd[2];      # File mode
say $passwd[7];      # File size
```

Súborový systém: Modul ‹File::stat›

Zmení správanie ‹stat› tak, aby vracala objekt:

```
use File::stat;

my $passwd = stat "/etc/passwd";
say $passwd->mode;
say $passwd->size;
```

Súborový systém: Operátory ‹-X›

```
-X [EXPR]
```

Predikát, ktorý otestuje nejakú vlastnosť súboru, používa ‹stat›.
Môže byť zadaný menom, alebo ako otvorený súbor.

Bez argumentu testuje ‹$_›.

```
-f FILE         # Same as (stat FILE)[2] & S_IFREG
-x FILE         # Same as (stat FILE)[2] & S_IXUSR
-s FILE         # Same as (stat FILE)[7]
```

Súborový systém: Operátory `-X`

`-r`, `-w`, `-x`, `-R`, `-W`, `-X`
: Prístup k súboru efektívnym (`-[rwx]`) či reálnym (`-[RWX]`) UID procesu.

`-o`, `-O`
: Vlastník súboru je zhodný s efektívnym či reálnym UID procesu.

`-e`
: Súbor existuje.

`-z`, `-s`
: Súbor je prázdny (***z**ero size*) alebo neprázdny.

`-s` vráti veľkosť súboru, `-z HANDLE` je teda rovnaké ako `!-s HANDLE`.

Súborový systém: Operátory `-X`

`-f` Bežný súbor

`-d` Adresár

`-l` Symbolický odkaz

> Používa `lstat()` namiesto `stat()`

`-p` Rúra

`-S` Soket

`-b`, `-c` Blokové alebo znakové zariadenie

`-t` Terminál

Súborový systém: Operátory `-X`

`-u`, `-g`, `-k`
: SetUID, SetGID a Sticky bit

`-T`, `-B`
: Binárny alebo textový súbor

> Používa veľmi hrubú heuristiku

`-M`, `-A`, `-C`
: Počet dní medzi spustením skriptu a časmi nastavenými na súbore (**M**odification, **A**ccess, **C**hange)

Súborový systém: Operátory ‹-X›

Každé volanie operátora nad reťazcom stojí systémové volanie.
Špeciálny otvorený súbor ‹_› zopakuje test na výsledku ‹stat› alebo iného testu:

```
stat FILE;

say "regular" if -f _;
say "directory" if -d _;
```

Perl má navyše syntaktický cukor pre podmienky, ktoré majú platiť súčasne:

```
-f -x FILE              # Same as -f FILE && -x FILE
```

Súborový systém: Zmena vlastností súboru

```
truncate EXPR, LENGTH
```

Zmení veľkosť otvoreného alebo pomenovaného súboru na ‹LENGTH›.

```
chmod MODE, LIST
chown UID, GID, LIST
```

Zmení práva, vlastníka a skupinu zadaných súborov.
‹MODE› zadávajte ideálne ako číslo, alebo použite ‹oct›.

```
chmod 0644, $fh;        # Okay.
chmod oct "0644", $fh;  # Also okay.
chmod "0644", $fh;      # Whoopsie, sets mode to <--w----r-T>!
```

Súborový systém: Zmena vlastností súboru

```
utime ATIME, MTIME, LIST
```

Zmení čas prístupu a úpravy zadaných súborov.

Modul ‹POSIX› obsahuje ďalšie rozširujúce funkcie.

Adresáre

Adresáre: Pracovný adresár (modul `Cwd`)

Pracovný adresár: Vlastnosť procesu.
Určuje, ako sa vyhodnotia relatívne cesty zadané iným systémovým volaniam.

```
use Cwd;
cwd
```

Vráti aktuálny adresár procesu.

```
getcwd
```

Ako `cwd`, ale navyše preloží symbolické odkazy.

Adresáre: Pracovný adresár

```
chdir [EXPR]
```

Zmení pracovný adresár procesu.
Argumentom môže byť reťazec alebo Directory Handle.

Ak nie je zadaný žiadny argument, použije `$ENV{HOME}`.

```
chdir "/etc";
open my $fh, "<", "passwd";     # Opens </etc/passwd>

chdir "/mnt/old/etc";
open my $fh2, "<", "passwd";    # Opens </mnt/old/etc/passwd>
```

Adresáre: Vytvorenie a zrušenie

```
mkdir [EXPR, [MODE]]
```

Vytvorí zadaný adresár. Ak nie je zadané meno, použije ‹$_›.

```
rmdir [FILENAME]
```

Zmaže zadaný **prázdny** adresár.

Adresáre: Modul ‹File::Path›

Práca s hlbokými adresárovými štruktúrami

```
use File::Path qw{make_path remove_tree};

make_path PATH₁, PATH₂, … [OPTIONS]
remove_tree PATH₁, PATH₂, … [OPTIONS]
```

Vytvoria alebo zrušia celú adresárovú štruktúru.
Nastavenia ‹OPTIONS› sú referencia na hash:

- ‹**mode**› Práva na nových adresároch
- ‹**safe**› Nebude sa pokúšať mazať neprístupné adresáre ‹keep_root›: Ponechá koreňový adresár každej zrušenej ‹PATH›.

Adresáre: Modul ‹File::Spec›

Práca s cestami

```
use File::Spec;

File::Spec->catfile(DIRECTORIES…, FILE);
File::Spec->catdir(DIRECTORIES…, FILE);
```

Vráti zloženú cestu pre súbor resp. adresár.

```
File::Spec->canonpath(PATH)
```

Vráti kanonickú cestu pre súbor.

```
File::Spec->rootdir
File::Spec->tmpdir
```

Vrátia systémové cesty pre koreňový adresár a adresár s dočasnými súbormi

Adresáre: Prechádzanie

```
opendir DIRHANDLE, EXPR
closedir DIRHANDLE
```

Obdoba ‹`open`› a ‹`close`› pre adresáre.

Ako pri súboroch, aj pri adresároch sa o zatvorenie postará Perl automaticky, keď zanikne posledná referencia na ‹`DIRHANDLE`›.

```
opendir my $dirh, "/etc"
    or die "/etc: $!\n";
```

Adresáre: Prechádzanie

```
readdir DIRHANDLE
```

V zoznamovom kontexte vráti (zostávajúce) položky v adresári.
V skalárnom vracia postupne položky adresára, po poslednej vráti ‹undef›.

```
while (readdir $dirh) {
    my $filename = File::Spec->catfile($dir, $_);
    say "file $filename" if -f $filename;
    say "dir  $filename" if -d $filename;
}
```

Položky obsahujú len názov objektu, nie celú cestu k nemu.

Adresáre: Prechádzanie

```
rewinddir DIRHANDLE
```

Vráti kurzor čítaného adresára na začiatok, takže ďalšie volanie `readdir` začne znova od prvého záznamu.

```
telldir DIRHANDLE
seekdir DIRHANDLE, POS
```

Vráti resp. nastaví pozíciu kurzora v čítanom adresári.

Adresáre: Glob operátor

```
glob EXPR
<*.c>
```

V zoznamovom kontexte vráti mená súborov v aktuálnom pracovnom adresári, ktoré vyhovujú zadanému vzoru (podobne ako shell).
V skalárnom kontexte vracia záznamy postupne, po poslednom `undef`.

```
glob "*.pl"             # Files ending with '.pl'
glob "x??"              # Files starting with 'x' and two more characters
glob "x[ab]"            # Either 'xa' or 'xb'
glob "ex{01,02}*"       # Files starting with 'ex01' or 'ex02'
glob "a* b*"            # Files starting with 'a' or 'b'
```

Ak vzor obsahuje **len** neprázdne `{...}` zátvorky, vráti expandované reťazce bez ohľadu na existenciu súborov (rovnako ako shell).

Adresáre: Glob operátor

Pozor na nejednoznačnosť `<X>` operátora!

Podľa `<X>` totiž môže znamenať `readline` alebo `glob`:

```
<FILE>          # Bareword → readline()
<$file>         # Scalar variable → readline()
< $file >       # Extra spaces? Too bad → glob()   (!!!)
<*.c>           # Expression → glob()
<$hash{key}>    # Expression → glob()
```

Ak nechcete riskovať, píšte `glob` explicitne.

Adresáre: Modul ‹File::Find›

```
use File::Find;

find(\&WANTED, DIRECTORIES…);
find(\%OPTIONS, DIRECTORIES…);      # $OPTIONS{wanted} = \&WANTED;
```

Rekurzívne prejde adresáre ‹DIRECTORIES…› a na každom súbore zavolá ‹WANTED›.
‹WANTED()› neberie žiadny argument; používa globálne premenné:

‹$_›	názov vrátený z ‹readdir›
‹$File::Find::dir›	aktuálny adresár
‹$File::Find::name›	názov vrátane cesty

Viac možností viď dokumentáciu ‹File::Find›.

Špeciálne literály

V zdrojovom kóde Perlu sa môže nachádzať šesť špeciálnych literálov:

- `__FILE__` Názov súboru
- `__LINE__` Aktuálne číslo riadka
- `__PACKAGE__` Názov modulu
- `__SUB__` Referencia na aktuálnu funkciu.

Špeciálne literály

`__END__` Označuje koniec skriptu, zvyšok súboru sa ignoruje.

`__DATA__` Označuje koniec skriptu, zvyšok súboru je dostupný ako súbor `DATA`. Takto je možné do skriptu pribaliť nejaké vstupy.

```
print while <DATA>;

__DATA__
1
2
3
```