برنامه نویسی به زبان

پرل

مرجع کامل

تالیف
مهندس هادی کیامرثی

تمام مثال های موجود در این کتاب با کامپیوتر تست شده اند تا از هر گونه خطا مبرا باشند با این حال ممکن است باز هم خطاهایی در آن وجود داشته باشد از کلیه خوانندگان این کتاب ، اساتید و دانشجویان محترم خواهشمندم برای مطلع کردن مولف از این خطا ها لطفا با ایمیل آدرس زیر تماس بگیرند

hadikiamarsi@gmail.com

فصل یازدهم

در این فصل مطالب زیر را خواهید آموخت

خواندن اطلاعات از ورودی

باز و بسته کردن فایل ها

تابع `open`

تابع `sysopen`

تابع `close`

خواندن و نوشتن در فایل ها

عملگر `<FILEHANDL>`

تابع `getc`

تابع `read`

تابع `print`

کپی کردن فایل ها

تغییر نام یک فایل

حذف یک فایل

تابع `tell`

تابع `seek`

دریافت اطلاعات فایل

خواندن اطلاعات از ورودی

برای خواندن یک رشته از صفحه کلید در زبان برنامه نویسی پرل از <STDIN> استفاده می نماییم برای آشنایی بیشتر با این درس به مثال زیر توجه نمایید

```
#!/usr/bin/perl

print "Please Enter Your name?\n";
$name = <STDIN>;
print "Your Name Is $name\n";
```

باز و بسته کردن فایل ها

برای باز کردن فایل ها در زبان برنامه نویسی پرل از دو تابع استفاده می شود که در زیر ساختار دستوری آن ها را مشاهده می نمایید

```
open FILEHANDLE, EXPR
open FILEHANDLE

sysopen FILEHANDLE, FILENAME, MODE, PERMS
sysopen FILEHANDLE, FILENAME, MODE
```

در اینجا نشان گر EXPR نوع حالت (خواندنی ، نوشتنی) باز کردن فایل می باشد

تابع open

برای باز کردن فایل **file.txt** بوسیله تابع open برای خواندن اطلاعات آن به روش زیر عمل می نماییم

```
open(DATA, "<file.txt");
```

در اینجا DATA یک اشاره گر فایل می باشد در مثال زیر اطلاعات فایل file.txt خط به خط خوانده می شود و در صفحه نمایش چاپ می گردد

```
#!/usr/bin/perl

open(DATA, "<file.txt") or die "Couldn't open file file.txt, $!";

while(<DATA>) {
   print "$_";
}
```

در مثال زیر فایل file.txt در حالت نوشتنی برای نوشتن اطلاعات در این فایل باز گشته است

```
open(DATA, ">file.txt") or die "Couldn't open file file.txt, $!";
```

لازم به ذکر است که در مثال بالا تمام اطلاعات قبلی فایل file.txt پاک می گردد

```
open(DATA, "+<file.txt"); or die "Couldn't open file file.txt, $!";
```

در مثال زیر یک فایل بدون اینکه اطلاعات قبلی آن باز گردد طوری باز می شود که بتوان اطلاعاتی در انتهای آن اضافه نمود

```
open DATA, "+>file.txt" or die "Couldn't open file file.txt, $!";
```

در مثال زیر فایل file.txt به حالت نوشتنی باز می شود به صورتی که اگر فایل file.txt وجود داشته باشد اجازه می دهد اطلاعات را به انتهای آن اضافه نماییم

```
open(DATA,">>file.txt") || die "Couldn't open file file.txt, $!";
```

در مثال زیر فایل file.txt به حالت نوشتنی باز می شود به صورتی که اگر فایل file.txt وجود داشته باشد اجازه می دهد اطلاعات را به انتهای آن اضافه نماییم

```
open(DATA,"+>>file.txt") || die "Couldn't open file file.txt, $!";
```

برای آشنایی بیشتر با حالت های باز کردن فایل به جدول زیر توجه نمایید

ردیف		شرح
1	< or r	یک فایل را فقط در حالت خواندنی باز می نماید
2	> or w	یک فایل را در حالت نوشتنی باز می نماید و اگر فایل موجود نباشد فایل را ایجاد می نماید
3	>> or a	یک فایل را طوری باز می نماید که بشود اطلاعات به انتهای آن اضافه نمود
4	+< or r+	یک فایل را در حالت

		خواندنی و نوشتنی باز می نماید
5	+> or w+	یک فایل را در حالت خواندنی و نوشتنی باز می نماید
6	+>> or a+	یک فایل را در حالت خواندنی و نوشتنی باز می نماید

تابع sysopen

تابع **sysopen** بسیار شبیه تابع **open()** می باشد با این تفاوت که گزینه هایی اضافی در آرگومان دارد برای تعیین نوع باز کردن فایل . برای آشنایی بیشتر با نحوه کاربرد این تابع به مثال زیر توجه نمایید

```
sysopen(DATA, "file.txt", O_RDWR);
```

Or to truncate the file before updating –

```
sysopen(DATA, "file.txt", O_RDWR|O_TRUNC );
```

میزان دسترسی به فایلی که توسط این تابع ایجاد می شود در سیستم عامل لینوکس و کلیه سیستم عامل های یونیکس بیس **0x666** می باشد

ردیف	آرگومان	عملکرد
1	O_RDWR	یک فایل را در حالت نوشتنی خواندی باز می نماید
2	O_RDONLY	یک فایل را در حالت خواندنی باز می نماید
3	O_WRONLY	یک فایل را در حالت نوشتنی باز می نماید
4	O_CREAT	یک فایل را ایجاد می کند
5	O_APPEND	یک فایل را طوری باز می نماید تا بتوان به انتهای فایل اطلاعات اضافه نمود

محتوای یک فایل را پاک می نماید	O_TRUNC	6
اگر فایل وجود داشته باشد هرگونه عملیاتی بر روی آن را متوقف می نماید	O_EXCL	7
یک فایل را طوری باز می نماید که به صورت بلاکی قابل دسترسی نباشد	O_NONBLOCK	8

تابع close

تابع **close** یک اشاره گر فایل را می بندد یا به عبارتی نابود می نماید

```
close FILEHANDLE
close
```

در مثال زیر DATA یک اشاره گر فایل می باشد که توسط تابع close نابود شده است

```
close(DATA) || die "Couldn't close file properly";
```

خواندن و نوشتن در فایل ها

وقتی یک فایل را باز می نمایید برای این است که اطلاعاتی را از فایل یا بخوانید یا اطلاعاتی در فایل بنویسید . در قسمت های بعدی بیشتر با روش های خواندن و نوشتن اطلاعات در فایل آشنا می شوید

عملگر <FILEHANDL>

همانطور که در قسمت های قبلی دیدید برای خواندن اطلاعات از صفحه کلید می توانید از <STDIN> مانند مثال زیر استفاده نمایید

```
#!/usr/bin/perl

print "What is your name?\n";
```

```
$name = <STDIN>;
print "Hello $name\n";
```

برای خواندن فایل ها نیز از همین روش می شود استفاده کرد برای روشن شدن موضوع به مثال زیر توجه نمایید

```
#!/usr/bin/perl

open(DATA,"<import.txt") or die "Can't open data";
@lines = <DATA>;
close(DATA);
```

تابع getc

تابع getc یک کاراکتر از فایل را می خواند

```
getc FILEHANDLE
getc
```

در مثال بالا باید به جای FILEHANDLE اشاره گر فایل را قرار دهید

تابع read

تابع read یک بلاک (گروه) از اطلاعات را از فایل می خواند

```
read FILEHANDLE, SCALAR, LENGTH, OFFSET
read FILEHANDLE, SCALAR, LENGTH
```

در دستور نحوی بالا FILEHANDLE اشاره گر فایل می باشد و SCALAR نقطه ای می باشد که خواندن اطلاعات فایل باید از آنجا شروع گردد در صورتی که گزینه OFFSET وجود نداشته باشد و LENGTH تعداد اطلاعاتی می باشد که باید از فایل خوانده شود و در صورتی که گزینه OFFSET وجود داشته باشد اطلاعات از بعد از OFFSET خوانده می شود .

تابع print

این تابع یک تابع چند کاربردی در زبان برنامه نویسی پرل می باشد یکی از کاربردهای آن که در مثال زیر نشان داده شده است نوشتن اطلاعات در فایل می باشد

```
print FILEHANDLE LIST
print LIST
print
```

یکی دیگر از کاربردهای این تابع که در فصل های قبل نیز با آن آشنا شدید چاپ اطلاعات در صفحه می باشد و در مثال زیر نشان داده شده است

```
print "Hello World!\n";
```

کپی کردن فایل ها

در مثال زیر نحوه کپی کردن فایل ها در زبان برنامه نویسی پرل نشان داده شده است

```
#!/usr/bin/perl

# Open file to read
open(DATA1, "<file1.txt");

# Open new file to write
open(DATA2, ">file2.txt");

# Copy data from one file to another.
while(<DATA1>) {
   print DATA2 $_;
}
close( DATA1 );
close( DATA2 );
```

تغییر نام یک فایل

برای تغییر نام یک فایل در زبان برنامه نویسی پرل از تابع پیش ساخته rename استفاده می شود . روش استفاده از این تابع در زیر آورده شده است

```
#!/usr/bin/perl

rename ("/usr/test/file1.txt", "/usr/test/file2.txt" );
```

این تابع دو آرگومان می پذیرد آرگومان اول نام و آدرس فایلی می باشد که قراره تغییر نام پیدا نماید و آرگومان دوم نام جدیدی می باشد که برای این فایل انتخاب کرده ایم .

حذف یک فایل

برای حذف یک فایل از روی حافظه کامپیوتر در زبان برنامه نویسی پرل از تابع پیش ساخته **unlink** استفاده می گردد روش استفاده از این تابع در زیر آورده شده است

```
#!/usr/bin/perl

unlink ("/usr/test/file1.txt");
```

تابع tell

تابع tell موقعیت اشاره گر فایل درهنگام خواندن اطلاعات را به بایت بر می گرداند دستور نحوی آن در زیر آورده شده است و بیاد داشته باشید FILEHANDLE اشاره گر فایل می باشد

```
tell FILEHANDLE
tell
```

تابع seek

تابع seek برای تعیین موقعیت اشاره گر فایل برای خواندن فایل بکار می رود

```
seek FILEHANDLE, POSITION, WHENCE
```

در دستور نحوی بالا FILEHANDLE اشاره گر فایل می باشد و POSITION موقعیت اشاره گر فایل برای خواندن اطلاعات می باشد و WHENCE موقعیتی است که از آنجا موقعیت اشاره گر خواندن فایل تعیین می گردد همانطور که در مثال زیر می بینید اگر WHENCE مقدار صفر داشته باشد تعیین موقعیت اشاره گر فایل از ابتدای فایل صورت می پذیرد

```
seek DATA, 256, 0;
```

دریافت اطلاعات فایل

در زبان برنامه نویسی پرل تعداد زیادی عملگر وجود دارد که بوسیله آن ها می توانید اطلاعاتی در مورد فایل ها بدست بیاورید در مثال زیر نحوه بکارگیری تعدادی از این عملگرها آورده شده است

```
#/usr/bin/perl

my $file = "/usr/test/file1.txt";
my (@description, $size);
if (-e $file) {
   push @description, 'binary' if (-B _);
   push @description, 'a socket' if (-S _);
   push @description, 'a text file' if (-T _);
   push @description, 'a block special file' if (-b _);
   push @description, 'a character special file' if (-c _);
   push @description, 'a directory' if (-d _);
   push @description, 'executable' if (-x _);
   push @description, (($size = -s _)) ? "$size bytes" : 'empty';
   print "$file is ", join(', ',@description),"\n";
}
```

در جدول زیر لیستی از عملگرهایی که بر روی فایل ها و دایرکتوری ها عمل می نمایند آورده شده است

ردیف	عملگر	توضیح
1	-B	آیا این فایل یک فایل باینری هست ؟
2	-C	آخرین زمان تغییر نود را بر می گرداند
3	-M	آخرین زمان تغییر فایل را بر می گرداند
4	-O	آیا صاحب فایل کاربر واقعی می باشد
5	-R	آیا این فایل برای یوزر واقعی قابل خواندن است ؟
6	-S	آیا یک فایل سوکت می باشد ؟
7	-T	آیا یک فایل متنی می باشد ؟
8	-W	آیا این فایل برای یوزر واقعی قابل نوشتن است ؟
9	-X	آیا برای یوزر واقعی این فایل قابل اجرا است ؟
10	-b	آیا فایل دارای بلاک خاص می باشد
11	-c	آیا فایل دارای کاراکترهای خاص می باشد
12	-d	آیا این فایل یک دایرکتوری هست ؟
13	-e	آیا فایل وجود دارد یا نه ؟
14	-f	آیا یک فایل ساده هست ؟
15	-g	آیا بیت setgid برای فایل وجود دارد یا نه ؟
16	-k	برای فایل تنظیم Sticky شده است یا نه ؟
17	-l	آیا این فایل یک فایل پیوند می باشد یا نه ؟
18	-o	آیا صاحب این فایل یک یوزر موثر است یا نه ؟
19	-p	آیا این فایل یک فایل لوله است ؟
20	-r	آیا این فایل قابل خواندن

می باشد		
اندازه فایل را بر می گرداند	-s	21
می TTYآیا این یک فایل باشد	-t	22
برای فایل تنظیم Setuid شده است	-u	23
آیا این فایل قبل نوشتن می باشد	-w	24
آیا این فایل قبل اجرا می باشد	-x	25
آیا اندازه فایل صفر می باشد ؟	-z	26
زمان آخرین دسترسی به فایل را مشخص می نماید	**-A**	27