Perl

CSE
오태호

요약문

이 문서는 Perl의 기본적인 사용법과 cgi프로그래밍에 응용하는 법에 대해 설명하고 있다.

1 Perl의 개요

Perl은 Practical Extraction and Report Language의 약자이다. Perl은 1986년 Larry Wall에 의해 만들어졌다. Larry Wall은 많은 양의 텍스트(text) 파일을 awk와 sed를 이용해 정리를 하는데 어려움을 느껴서 프로그래밍을 하기 시작했는데 이것이 Perl이다. Perl이 요즘 유용하게 쓰이는 이유는 CGI프로그래밍을 할 때 여러가지 텍스트파일의 처리가 필요한데 Perl이 텍스트파일의 처리에 아주 강력한 언어이기 때문이다. 또한 Perl은 프로그램 코드를 해석(interpret)하기 때문에 번역(compile)을 하는 불편이 없고 프로그램도 쉽게 수정할 수 있다. 하지만 번역된 코드보다 느리다는 단점이 있는데 Perl용 번역기(compiler)도 있으니 크게 걱정할 필요는 없다.

2 Perl의 문법

2.1 아주 간단한 Perl 프로그램

```
#!/usr/local/bin/perl
print "Hello World\n";
# 이건 주석입니다.
```

이 프로그램의 실행결과는 다음과 같다.

```
Hello World
```

이 코드를 보면 Perl 프로그램은 C와 상당히 유사하다는 것을 알 수 있다. 첫번째 줄은 일반 쉘 스크립트(shell script)와 같이 사용할 스크립트의 언어를 정해준다. [1] 두번째 줄은 C의 printf함수와 흡사하다. \n은 개행문자이다. Perl이 C와 다른 한가지는 Perl에서는 함수를 쓸 때 괄호가 없어도 된다는 점이다. 하지만 원한다면 괄호를 써도 된다. 아래와 같이 해도 위와 동일한 결과를 얻을 수 있다. 이 글을 읽다보면 알 수 있겠지만 함수사용에 필요한 인자도 경우에 따라서 생략할 수도 있다. 생략을 하면 Perl이 초기값(default)으로 어떤 인자를 사용한다. 그리고 Perl에서는 #이 나오면 그 뒤에 나오는 한 행이 주석으로 처리된다.

[1] 일반적으로 /usr/local/bin/perl 또는 /usr/bin/perl 이다.

```
#!/usr/local/bin/perl
print("Hello World\n");
```

2.2 자료형(Data Type)

Perl은 C에 비해 자료형의 개수가 아주 적다. C의 unsigned long이나 int나 모두 같은 숫자로 취급된다. 또한 Perl에서는 어떤 변수를 사용하기 위해 사용전에 선언을 할 필요가 없다. Perl은 이와 같이 C에 비해서 융통성이 크다.

2.2.1 스칼라(Scalar)

모든 숫자형태의 변수와 문자열은 스칼라형이다. 스칼라형은 변수의 앞에 $가 붙는다. 다음 프로그램은 Perl의 강력한 융통성을 단적으로 보여준다.

```
#!/usr/local/bin/perl
$a=2;
$b=6;
$c=$a.$b;
$d=$c/2;
print $d;
```

결과

```
13
```

$a와 $b라는 이름의 변수에 2와 6을 넣는다. '.'은 연산자의 일종으로 두개의 문자열을 하나로 합치는 연산자이다. 그런데 여기에서는 숫자에서 사용하고 있다. 하지만 Perl은 숫자를 문자로 인식해서 연산을 수행해서 결과를 $c에 넣는다. $c를 2로 나눠주면 $c를 숫자로 인식해서 2로 나눠주고 $d에 넣는다.

이와같이 Perl은 숫자와 문자열을 같은 스칼라형으로 인식한다.

2.2.2 배열(Array)

배열형은 스칼라형의 변수가 여러개 모인 것이다. 배열형의 변수는 앞에 @가 붙는다. 그리고 스칼라형의 데이타를 참조하려면 $와 [ ]를 사용해서 참조한다.

```
#!/usr/local/bin/perl
@ids=("ohhara","eliot","iblion");
@numbers=(100,122,5);
@mix=("ohhara",10,"eliot");
print "$ids[0] $ids[1] $ids[2]\n";
print "$numbers[0] $numbers[1] $numbers[2]\n";
print "$mix[0] $mix[1] $mix[2]\n";
```

결과

```
ohhara eliot iblion
100 122 5
ohhara 10 eliot
```

2.2.3 관계 배열(Associative Array)

관계 배열형은 배열과 비슷하지만 한가지가 다르다. 배열형은 스칼라형의 데이타를 참조할 때 인덱스(index)번호를 [ ] 사이에 넣어서 참조하지만 관계 배열형은 { }사이에 키(key)를 넣어서 참조한다. 관계 배열형 변수 앞에는 %가 붙는다. 스칼라형의 데이타를 참조하려면 $와 { }를 사용해서 참조한다. 이것을 이용하면 간단한 데이타베이스에도 응용이 가능하다.

```
#!/usr/local/bin/perl
%students=("ohhara",9725018,"eliot",9725002,"iblion",9725035);
print "$students{'ohhara'} $students{'eliot'} $students{'iblion'}\n";
```

결과

```
9725018 9725002 9725035
```

2.2.4 파일 핸들(File Handle)

파일 핸들은 엄밀하게 말하자면 자료형이 아니다. 하지만 파일 핸들은 변수처럼 행동할 때가 많다. 그래서 자료형과 같이 설명하겠다.

파일 핸들은 이미 정의되어 있는 것이 3개가 있다. STDIN[2] ,STDOUT[3] ,STDERR[4] 이다. 이것은 C와 용어도 같고 의미도 거의 비슷하다. 새로운 파일 핸들을 만들때 open을 쓰면 쉽게 만들 수 있다.

```
#!/usr/local/bin/perl
open(INFILE,"<ohhara.in");
open(OUTFILE,">ohhara.out");
while(<INFILE>)
{
   print OUTFILE;
}
close(INFILE);
close(OUTFILE);
```

이 프로그램을 실행시키면 ohhara.in에 있는 내용이 ohhara.out으로 복사된다. 일단 크게보면 이 프로그램은 open을 이용해서 파일을 열고 close를 이용해서 파일을 닫는다. 이것은 C의 fopen,fclose와 유사하다. open을 이용해서 ohhara.in파일을 읽기전용(<)으로 연다. 그리고 ohhara.out파일을 쓰기전용(>)으로 연다. 그리고 각각의 파일은 INFILE 파일 핸들과 OUTFILE 파일 핸들을 이용해서 사용한다.[5] while은 나중에 설명할 내용인데 간단히 설명하자면 ( )안의 코드가 참일동안 계속해서 { }안의 코드를 반복하라는 뜻이다. <>는 파일 핸들을 다루는 연산자의 일종이다. <>는 <>안에 있는 파일 핸들에서 한줄을 읽어들인다. 그러므로 while은 ohhara.in파일을 끝까지 읽어들일 때까지 반복된다. 여기서 이상한 점을 하나 발견할 수 있다. INFILE로 파일을 한줄씩 읽어들이는데 읽어서 아무데도 저장하지 않았고 프린트를 할 때는 프린트를 할 파일 핸들은 정해줬지만 무엇을 프린트해야 하는지는

[2] 일반적으로 키보드이다.

[3] 일반적으로 모니터이다. 웹인 경우에는 브라우저이다.

[4] 일반적으로 모니터 또는 기록(log)파일이다.

[5] 이 개념은 C를 해본 사람에게 익숙한 개념이다.

정해주지 않았다. <>를 이용해서 파일을 한줄씩 읽어들이면 $_라는 특수한 변수에 저장된다. 그리고 위에서 잠깐 언급을 한 것과 같이 함수에서 인자가 부족하면 Perl이 알아서 초기값을 사용한다고 했다. print함수에서 무엇을 프린트해야 하는지 인자가 하나 부족한데 이럴 경우에는 $_를 대신 인자로 사용한다. 즉 위의 프로그램에서 print OUTFILE;는 print OUTFILE $_;과 동일한 의미를 가진다.[6] 그러므로 ohhara.in에서 한줄씩 파일을 읽어들여서 그대로 ohhara.out에 쓰게 된다.

2.3 흐름 제어(Flow Control)

2.3.1 ||와 &&

||는 or이고 &&는 and이다. ||와 &&는 C에서의 사용법과 거의 비슷하다. ||의 왼쪽에 있는 조건식과 오른쪽에 있는 조건식중 하나라도 참이면 참을 반환(return)한다. &&는 왼쪽에 있는 조건식과 오른쪽에 있는 조건식이 모두 참이면 참을 반환한다. 이것은 단순히 if같은 곳에만 쓰이는게 아니라 일반적인 곳에도 많이 쓰인다. ||에서 앞에 있는 코드가 참이면 뒤에 있는 코드는 실행시키지 않고 참을 반환한다. &&에서 앞에 있는 코드가 거짓이면 뒤에있는 코드는 실행시키지 않고 거짓을 반환한다. 정리해서 말하면 ||의 앞에 있는 코드가 거짓이면 뒤에 있는 코드를 실행시키고 &&의 앞에 있는 코드가 참이면 뒤에 있는 코드를 실행시킨다.

```
#!/usr/local/bin/perl
$a=10;
$b=20;
$c=30;
$d=40;
$a > $b || print "$a\n";
$a < $b || print "$b\n";
$c > $d && print "$c\n";
$c < $d && print "$d\n";
```

결과

```
10
40
```

2.3.2 if와 unless

if는 C의 for와 거의 비슷하다. if 다음에 있는 ( )안의 코드가 참이면 { }안의 코드를 실행시킨다. 그리고 { }안의 코드가 한 문장일 때[7] 간단한 다른 형태로 쓸 수도 있다. if문의 뒤에 있는 코드가 참이면 앞에 있는 코드를 실행시키는 형태로 쓸 수 있다. { }의 뒤에 else를 쓰고 { }를 하면 if다음에 있는 ( )안의 코드가 거짓일 때 else다음에 있는 { }안의 코드를 실행시킨다. elsif는 else if라고 생각 하면 된다.[8] unless는 if와 모든게 동일하지만 ( )안의 코드가 거짓일 때 { }안의 코드를 실행시킨다.

---

[6] print함수에서 파일 핸들을 입력하지 않으면 일반적으로 STDOUT을 사용한다.

[7] 문장은 ;로 구분된다.

[8] 단, else if라고 쓰면 안된다. elsif라고 써야 된다.

```
#!/usr/local/bin/perl
$foo=1000;
$bar=1000;
if($foo>500)
{
   $foo=500;
}
unless($bar>500)
{
   $bar=500;
}
print "$foo $bar";
```

결과

```
500 1000
```

이 프로그램은 다음과 같이 해도 동일하다.

```
#!/usr/local/bin/perl
$foo=1000;
$bar=1000;
$foo=500 if $foo>500;
$bar=500 unless $bar>500;
print "$foo $bar";
```

2.3.3 while과 until

while은 앞에서 언급된 것과 같이 ( )안의 코드가 참일 동안 { }안의 코드를 반복한다. while도 if문과 { }안에 문장이 하나일 때 간단한 형태로 쓸 수 있다. until은 while과 모든게 동일하지만 ( )안의 코드가 거짓일 동안 { }안의 코드를 반복한다.

```
#!/usr/local/bin/perl
$i=1;
$total=0;
while($i<=100)
{
   $total+=$i; # $total=$total+$i;
   $i++;       # $i=$i+1;
}
print "$total\n";

$i=1;
$total=0;
until($i>100)
{
   $total+=$i;
```

```
    $i++;
}
print "$total\n";
```

결과

```
5050
5050
```

이 프로그램은 1부터 100까지 더한다. 위에 while을 쓴 부분과 until을 쓴 부분은 같은 동작을 한다.

2.3.4 for와 foreach

for는 C의 for와 거의 비슷하다. for는 ( )안에 ;으로 3부분으로 나누어 지고 뒤에 { }로 둘러싸인 부분이 있다. for에서 ( )안의 첫번째 부분은 { }안의 코드를 실행시키기 전에 실행되는 코드이다. 두번째 부분은 조건식인데 이 부분이 참일 동안 { }안의 코드를 반복한다. 세번째 부분은 { }안의 코드를 실행시킬때마다 실행되는 코드이다.

foreach는 스칼라형 변수가 있고 그 뒤의 ( )안에 배열이 있고 그 뒤에 { }가 있다. 배열에 있는 스칼라를 하나씩 스칼라형 변수에 넣으면서 { }안의 코드를 실행시킨다.

```
#!/usr/local/bin/perl
for($i=1;$i<=100;$i++)
{
    $total+=$i;
}
print "$total\n";
```

결과

```
5050
```

위의 while과 until에서 나온 예제와 같은 동작을 한다.

```
#!/usr/local/bin/perl
@ids=("ohhara","eliot","iblion");
foreach $id (@ids)
{
    print "$id\n";
}
```

결과

```
ohhara
eliot
iblion
```

2.3.5 서브루틴(Subroutine)

서브루틴은 C의 함수와 거의 비슷하다. 서브루틴을 정의할 때는 sub로 정의하고 호출할 때는 앞에 &를 붙여서 호출한다. 함수 인자는 @_[9] 라는 특수한 배열을 사용한다. 반환값은 서브루틴의 제일 마지막에 있는 변수이다.

```
#!/usr/local/bin/perl
$a=&printnumber(10,20);
print "$a\n";
sub printnumber
{
   print "$_[0] $_[1]\n";
   $_[0]+$_[1];
}
```

결과

```
10 20
30
```

2.4 패턴(Pattern)

Perl은 텍스트의 패턴처리가 상당히 강력하다. 그래서 모두 다 설명하기는 너무 많다. 자주 쓰는 것만 몇개 소개한다. 그리고 표를 참조하기 바란다.

Assertion	Matches
^	문자열의 시작
$	문자열의 끝
.	개행문자를 제외한 임의의 문자
[ ]	[ ]안에 들어있는 모든 문자
*	바로 앞에 있는 문자의 0번이상 반복
+	바로 앞에 있는 문자의 1번이상 반복
?	바로 앞에 있는 문자가 0~1번 나옴
\d	숫자
\D	숫자가 아닌 것
\w	숫자나 알파벳
\W	숫자나 알파벳이 아닌 것
{n}	바로 앞에 있는 문자의 n번 반복
{n,}	바로 앞에 있는 문자의 n번이상 반복
{n,m}	바로 앞에 있는 문자의 n~m번 반복

[9]@_에 있는 배열안의 스칼라를 참조할 때는 $_[0] $_[1] 같은 방법으로 참조한다.

```
#!/usr/local/bin/perl
$house="henhouse";
if($house=~/^hen/)
{
   print "$house\n";
}
$house=~tr/h/p/;
print "$house\n";
$house=~s/use/abc/;
print "$house\n";
```

결과

```
henhouse
penpouse
penpoabc
```

=~연산자는 왼쪽에 있는 문자열에서 어떤 특정한 패턴을 찾거나 찾은 것을 이용해 치환을 할 때 아주 자주 사용되는 연산자이다. 첫번째 사용에서는 $house가 hen으로 시작되는 문자열인지 확인한다. 두번째 사용에서는 $house에 있는 h라는 문자를 p라는 문자로 치환한다. 세번째 사용에서는 $house에 있는 use라는 문자열을 abc라는 문자로 치환한다.

3 방명록 소스 분석

```
#!/usr/bin/perl

# 이 방명록은 Matt씨가 만든 것이다.
# 이 방명록은 http://www.worldwidemart.com/scripts에서 가져온 것이다.
# 이 방명록에서 코드는 바꾼 것이 없고 단지 주석만 제가 작성했다.

# 이 부분은 방명록에 필요한 여러가지 환경을 설정해 주는 곳이다.

$guestbookurl = "http://your.host.com/~yourname/guestbook.html";
$guestbookreal = "/home/yourname/public_html/guestbook.html";
$guestlog = "/home/yourname/public_html/guestlog.html";
$cgiurl = "http://your.host.com/cgi-bin/guestbook.pl";
$date_command = "/usr/bin/date";

$mail = 0;              # 1 = Yes; 0 = No
$uselog = 1;            # 1 = Yes; 0 = No
$linkmail = 1;          # 1 = Yes; 0 = No
$separator = 1;         # 1 = <hr>; 0 = <p>
$redirection = 0;       # 1 = Yes; 0 = No
$entry_order = 1;       # 1 = Newest entries added first;
```

```
                         # 0 = Newest Entries added last.
$remote_mail = 0;        # 1 = Yes; 0 = No
$allow_html = 1;         # 1 = Yes; 0 = No
$line_breaks = 0;# 1 = Yes; 0 = No

$mailprog = '/usr/lib/sendmail';
$recipient = 'you@your.com';

# ``으로 문자열이 둘러싸여 있으면 ``안에 있는 명령어를 시스템상에서 실행시킨
# 결과값이 반환된다.
# $date와 $shortdate에는 각각 다른 형태의 시간이 기록된다.
# +뒤에 있는 내용은 도움말(manual page)에 자세히 나와 있다.
# chop함수는 문자열의 제일 끝에 있는 문자를 제거하는 함수이다.
# 여기서는 개행문자를 없애기 위해 사용했다.

$date = `$date_command +"%A, %B %d, %Y at %T (%Z)"`; chop($date);
$shortdate = `$date_command +"%D %T %Z"`; chop($shortdate);

# $ENV는 여러가지 환경변수를 저장하고 있는 특수한 관계 배열이다.
# 아래와 같이 하면 form의 method가 post인 경우 form에 입력한 내용이 $buffer
# 에 길게 기록된다.
# read(a,b,c)가 있으면 a에서 c만큼 읽어서 b에 저장하는 함수이다.

read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});

# split함수는 어떤 패턴을 기준으로 문자열을 잘라서 배열로 만드는 함수이다.
# 월드 와이드 웹(World Wide Web)을 써 본 사람은 가끔 URL로 이런 형식의 정보
# 가 전해지는 것을 봤을 것이다.(id=ohhara&name=Taeho+Oh)
# 여기에서 일단 &단위로 끊어서 배열을 만드는 것이다.

@pairs = split(/&/, $buffer);

foreach $pair (@pairs) { # @pairs 배열에 있는 것을 하나하나 $pair에 넣는다.
   ($name, $value) = split(/=/, $pair); # $pair를 =로 끊어서 배열을 만든다.

   # 일반적으로 form에 입력할 때 띄워쓰기를 하면 +가 첨가된다.
   # 아래는 그런 +를 공백으로 만들어 준다.

   $value =~ tr/+/ /;

   # 특수문자 같은 것들은 %2C 같이 %와 16진수 코드로 전해진다.
   # 아래를 보면 %가 앞에 있고 알파벳이나 숫자가 두번 반복되는 것이 있으면
   # $1에 일단 저장을 하고 (치환에서 괄호에 둘러싸인 것은 $1 $2이런 식으로
   # 차례대로 대입된다.) 이것을 아스키(ASCII)문자로 변환한다.
```

```
   # hex함수는 16진수를 10진수로 바꾸는 함수이고 pack에 첫번째 인자를 "C"를
   # 주면 두번째 인자를 아스키코드로 인식해 문자로 변환한다.
   # 그리고 뒤에 있는 e와 g는 e는 아까 pack이 포함되어 있는 곳을 계산하라는
   # 뜻이고 g는 $value에 있는 모든 문자열에 대해 적용시키라는 뜻이다.
   # g를 생략하게 되면 치환은 한번만 일어난다.

   $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;

   # 입력한 내용중에 html형식의 주석이 있으면 모두 제거한다.

   $value =~ s/<!--(.|\n)*-->//g;

   # html을 허용하지 않은 경우에는 html tag를 모두 제거한다.

   if ($allow_html != 1) {
      $value =~ s/<([^>]|\n)*>//g;
   }

   # $name을 키값으로 $value를 구할 수 있는 %FORM이라는 관계 배열을 만든다.
   # 여기서 $name는 html의 form에서 name field에 입력한 값이고
   # $value는 사용자가 입력한 값이다.

   $FORM{$name} = $value;
}


# comments에 아무것도 입력하지 않았으면 no_comments라는 서브루틴을
# 실행시킨다.

&no_comments unless $FORM{'comments'};

# realname에 아무것도 입력하지 않았으면 no_name이라는 서브루틴을
# 실행시킨다.

&no_name unless $FORM{'realname'};

# $guestbookreal을 읽기전용으로 open하고 실패하면 오류(error)를 출력한다.
# 여기에서 출력한 오류는 웹 서버(web server)의 기록파일에 기록된다.
# 참고로 $!는 오류 숫자(error numver)를 나타내는 특수한 변수이다.

open (FILE,"$guestbookreal") || die "Can't Open $guestbookreal: $!\n";

# <>는 한 라인씩 읽어들이기도 하지만 다음과 같이 사용하면 파일을 한번에
# 읽어서 @LINES 배열에 한줄씩 기록한다.
```

```
@LINES=<FILE>;

# 파일을 닫는다.

close(FILE);

# 이런식으로 사용하면 @LINES 배열에 있는 줄의 개수가 $SIZE에 기록된다.

$SIZE=@LINES;

# $guestbookreal을 쓰기전용으로 연다. 실패하면 error를 출력한다.

open (GUEST,">$guestbookreal") || die "Can't Open $guestbookreal: $!\n";

for ($i=0;$i<=$SIZE;$i++) { # $guestbookreal의 줄수만큼 반복한다.
   $_=$LINES[$i]; # $_에 @LINES를 한줄씩 대입한다.

   # <!--begin-->이 포함되어 있는 줄을 만나면 if를 실행시키고
   # 그렇지 않으면 그대로 파일에 프린트한다.(else는 한참 아래에 있다.)

   if (/<!--begin-->/) {

      # 최근에 쓴 글이 제일 위에 올라오도록 되어 있으면 <!--begin-->을
      # 파일에 프린트한다.
      # 참고로 eq는 문자열 두개가 같으면 참을 반환하는 연산자이다.

      if ($entry_order eq '1') {
         print GUEST "<!--begin-->\n";
      }

      # 엔터(개행문자)를 입력한 것을 인식하게 되어 있으면 개행문자를 <br>로
      # 치환한다.

      if ($line_breaks == 1) {
         $FORM{'comments'} =~ s/\cM\n/<br>\n/g;
      }

      # 사용자가 입력한 글의 내용을 파일에 프린트한다.

      print GUEST "<b>$FORM{'comments'}</b><br>\n";

      # 사용자가 URL을 입력했으면 그 URL로 연결(link)을 만들어 파일에
      # 프린트한다.
```

```
# URL을 입력하지 않았으면 realname만 파일에 프린트한다.

if ($FORM{'url'}) {
   print GUEST "<a href=\"$FORM{'url'}\">$FORM{'realname'}</a>";
}
else {
   print GUEST "$FORM{'realname'}";
}

# username(e-mail address)을 입력했으면 if를 실행시킨다.

if ( $FORM{'username'} ){

   # email을 연결시키도록 되어 있으면 연결을 만들어 파일에 프린트한다.
   # 그렇게 되어 있지 않으면 email만 파일에 프린트한다.

   if ($linkmail eq '1') {
      print GUEST " \&lt;<a href=\"mailto:$FORM{'username'}\">";
      print GUEST "$FORM{'username'}</a>\&gt;";
   }
   else {
      print GUEST " &lt;$FORM{'username'}&gt;";
   }
}

# city state country등을 입력했으면 파일에 프린트한다.

print GUEST "<br>\n";

if ( $FORM{'city'} ){
   print GUEST "$FORM{'city'},";
}

if ( $FORM{'state'} ){
   print GUEST " $FORM{'state'}";
}

if ( $FORM{'country'} ){
   print GUEST " $FORM{'country'}";
}

# 방명록 데이타를 <hr>로 나누게 되어 있으면 <hr>로 나눠서 파일에
# 프린트한다.
# 그렇지 않으면 <p>로 나누어 프린트한다.
```

```
        if ($separator eq '1') {
            print GUEST " - $date<hr>\n\n";
        }
        else {
            print GUEST " - $date<p>\n\n";
        }

        # 최근에 올린 글이 제일 밑에 가게 되어 있으면 끝에 <!--begin-->을
        # 파일에 프린트한다.

        if ($entry_order eq '0') {
            print GUEST "<!--begin-->\n";
        }

    }
    else {
        print GUEST $_;
    }
}

# 파일을 닫는다.

close (GUEST);

# 기록파일을 기록하게 되어 있으면 log 서브루틴을 실행시킨다.

if ($uselog eq '1') {
    &log('entry');
}

# 방명록을 만든 사람에게 사용자가 쓴 글을 편지로 보내도록 되어 있을 때 if를
# 실행한다.

if ($mail eq '1') {

    # 아래와 같이 하면 MAIL에 프린트한 내용이 $mailprog의 인자로 전달된다.
    # 어쩌다 보니 설명이 조금 애매하게 되었는데 이건 UNIX의 pipe개념과 같다고
    # 생각하면 무리가 없다. 이해가 잘 가지 않으면 UNIX에서 pipe를 참고하면
    # 된다.

    open (MAIL, "|$mailprog $recipient") || die "Can't open $mailprog!\n";

    # 사용자가 form에 입력한 내용을 파일에 프린트한다.
```

```
   print MAIL "Reply-to: $FORM{'username'} ($FORM{'realname'})\n";
   print MAIL "From: $FORM{'username'} ($FORM{'realname'})\n";
   print MAIL "Subject: Entry to Guestbook\n\n";
   print MAIL "You have a new entry in your guestbook:\n\n";
   print MAIL "------------------------------------------------------\n";
   print MAIL "$FORM{'comments'}\n";
   print MAIL "$FORM{'realname'}";

   if ( $FORM{'username'} ){
      print MAIL " <$FORM{'username'}>";
   }

   print MAIL "\n";

   if ( $FORM{'city'} ){
      print MAIL "$FORM{'city'},";
   }

   if ( $FORM{'state'} ){
      print MAIL " $FORM{'state'}";
   }

   if ( $FORM{'country'} ){
      print MAIL " $FORM{'country'}";
   }

   print MAIL " - $date\n";
   print MAIL "------------------------------------------------------\n";

   # 파일을 닫는다.

   close (MAIL);
}

# 사용자가 입력한 내용을 편지로 보내도록 되어 있을때 if를 실행한다.

if ($remote_mail eq '1' && $FORM{'username'}) {
   open (MAIL, "|$mailprog -t") || die "Can't open $mailprog!\n";

   print MAIL "To: $FORM{'username'}\n";
   print MAIL "From: $recipient\n";
   print MAIL "Subject: Entry to Guestbook\n\n";
   print MAIL "Thank you for adding to my guestbook.\n\n";
```

```
    print MAIL "-------------------------------------------------------\n";
    print MAIL "$FORM{'comments'}\n";
    print MAIL "$FORM{'realname'}";

    if ( $FORM{'username'} ){
       print MAIL " <$FORM{'username'}>";
    }

    print MAIL "\n";

    if ( $FORM{'city'} ){
       print MAIL "$FORM{'city'},";
    }

    if ( $FORM{'state'} ){
       print MAIL " $FORM{'state'}";
    }

    if ( $FORM{'country'} ){
      print MAIL " $FORM{'country'}";
    }

    print MAIL " - $date\n";
    print MAIL "-------------------------------------------------------\n";

    close (MAIL);
}

# 사용자가 입력한 내용을 감사메세지와 함께 보여주지 않게 되어 있으면 if를
# 그렇지 않으면 no_redirection 서브루틴을 실행한다.

if ($redirection eq '1') {
    print "Location: $guestbookurl\n\n";
}
else {
    &no_redirection;
}

sub no_comments {

    # comments를 입력하지 않았다는 메세지를 출력한다.
    # Content-type: text/html\n\n 에서 \n은 반드시 두번을 넣어줘야 된다.
    # 그래야 웹서버에서 정상적으로 이것을 인식할 수 있다.
```

```
   print "Content-type: text/html\n\n";
   print "<html><head><title>No Comments</title></head>\n";
   print "<body><h1>Your Comments appear to be blank</h1>\n";
   print "The comment section in the guestbook fillout form appears\n";
   print "to be blank and therefore the Guestbook Addition was not\n";
   print "added.  Please enter your comments below.<p>\n";
   print "<form method=POST action=\"$cgiurl\">\n";
   print "Your Name:<input type=text name=\"realname\" size=30 ";
   print "value=\"$FORM{'realname'}\"><br>\n";
   print "E-Mail: <input type=text name=\"username\"";
   print "value=\"$FORM{'username'}\" size=40><br>\n";
   print "City: <input type=text name=\"city\" value=\"$FORM{'city'}\" ";
   print "size=15>, State: <input type=text name=\"state\" ";
   print "value=\"$FORM{'state'}\" size=15> Country: <input type=text ";
   print "name=\"country\" value=\"$FORM{'country'}\" size=15><p>\n";
   print "Comments:<br>\n";
   print "<textarea name=\"comments\" COLS=60 ROWS=4></textarea><p>\n";
   print "<input type=submit> * <input type=reset></form><hr>\n";
   print "Return to the <a href=\"$guestbookurl\">Guestbook</a>.";
   print "\n</body></html>\n";

   if ($uselog eq '1') {
      &log('no_comments');
   }

   exit;
}

sub no_name {

   # name을 입력하지 않았다는 메세지를 출력한다.

   print "Content-type: text/html\n\n";
   print "<html><head><title>No Name</title></head>\n";
   print "<body><h1>Your Name appears to be blank</h1>\n";
   print "The Name Section in the guestbook fillout form appears to\n";
   print "be blank and therefore your entry to the guestbook was not\n";
   print "added.  Please add your name in the blank below.<p>\n";
   print "<form method=POST action=\"$cgiurl\">\n";
   print "Your Name:<input type=text name=\"realname\" size=30><br>\n";
   print "E-Mail: <input type=text name=\"username\"";
   print " value=\"$FORM{'username'}\" size=40><br>\n";
   print "City: <input type=text name=\"city\" value=\"$FORM{'city'}\" ";
   print "size=15>, State: <input type=text name=\"state\" ";
```

```
   print "value=\"$FORM{'state'}\" size=2> Country: <input type=text ";
   print "value=USA name=\"country\" value=\"$FORM{'country'}\" ";
   print "size=15><p>\n";
   print "Comments have been retained.<p>\n";
   print "<input type=hidden name=\"comments\" ";
   print "value=\"$FORM{'comments'}\">\n";
   print "<input type=submit> * <input type=reset><hr>\n";
   print "Return to the <a href=\"$guestbookurl\">Guestbook</a>.";
   print "\n</body></html>\n";

   if ($uselog eq '1') {
      &log('no_name');
   }

   exit;
}

sub log {

   # 이 함수의 첫번째 인자로 넘어온 것을 $log_type에 대입한다.

   $log_type = $_[0];

   # 아래와 같이 파일을 열면 $guestlog에 프린트할 때 파일 뒤에 추가한다.

   open (LOG, ">>$guestlog");

   # %ENV는 위에서 설명했던 것과 같이 특수한 변수이다.
   # $ENV{'REMOTE_HOST'} 라고 하면 접속한 곳의 도메인 네임을 반환한다.

   if ($log_type eq 'entry') {
      print LOG "$ENV{'REMOTE_HOST'} - [$shortdate]<br>\n";
   }
   elsif ($log_type eq 'no_name') {
      print LOG "$ENV{'REMOTE_HOST'} - [$shortdate] - ERR: No Name<br>\n";
   }
   elsif ($log_type eq 'no_comments') {
      print LOG "$ENV{'REMOTE_HOST'} - [$shortdate] - ERR: No ";
      print LOG "Comments<br>\n";
   }
}

sub no_redirection {
```

```
# 감사메세지를 출력하고 사용자가 입력한 내용을 보여준다.

print "Content-Type: text/html\n\n";
print "<html><head><title>Thank You</title></head>\n";
print "<body><h1>Thank You For Signing The Guestbook</h1>\n";

print "Thank you for filling in the guestbook.  Your entry has\n";
print "been added to the guestbook.<hr>\n";
print "Here is what you submitted:<p>\n";
print "<b>$FORM{'comments'}</b><br>\n";

if ($FORM{'url'}) {
   print "<a href=\"$FORM{'url'}\">$FORM{'realname'}</a>";
}
else {
   print "$FORM{'realname'}";
}

if ( $FORM{'username'} ){
   if ($linkmail eq '1') {
      print " &lt;<a href=\"mailto:$FORM{'username'}\">";
      print "$FORM{'username'}</a>&gt;";
   }
   else {
      print " &lt;$FORM{'username'}&gt;";
   }
}

print "<br>\n";

if ( $FORM{'city'} ){
   print "$FORM{'city'},";
}

if ( $FORM{'state'} ){
   print " $FORM{'state'}";
}

if ( $FORM{'country'} ){
   print " $FORM{'country'}";
}

print " - $date<p>\n";
```

```
    print "<hr>\n";
    print "<a href=\"$guestbookurl\">Back to the Guestbook</a>\n";
    print "- You may need to reload it when you get there to see your\n";
    print "entry.\n";
    print "</body></html>\n";

    exit;
}
```