CS306: Introduction to Perl

Section #7: Special Variables

U. of Alabama at Birmingham
Dept. of Computer & Information Sciences

Section 7: Special Variables

The default variable
Parameter list special variables
I/O-related special variables
Environment special variables

The Default Variable - $_

- Perl's most useful and important special variable
- Also known as the "$ARG" variable
  - $ARG does work, but no one uses it. Use $_
- $_ is the default input space. When Perl expects a place to put input and you don't give it any, it will use $_
- Many functions will use $_ if they expect a variable and you don't provide one.

$_ Continued

- ```
  foreach $name (@names) {
      print "$name\n";
  }
  ```

- ```
  foreach (@names) {   # no var given; uses $_
      print $_;
  }
  ```

$_ Continued

- We also said some functions will use $_ in the absence of provided data
- print() is one of these

- ```
  for (@names) {  # uses $_
      print;      # also uses $_ since no var given
  }
  ```


$_ Continued

- ```
  while ($line = <>) {
      print "The input was $line;\n";
  }
  ```

- ```
  while (<>) {
      print "The line was $_\n";
  }
  ```


$_ Continued

- chomp() is another function that will use $_
- ```
  while (<FH>) {
      chomp;
      print "No more newline in $_.";
  }
  ```
- Many functions that expect a list or scalar will use $_. The Functions chapter of the book will note when a function will use $_

The Parameter List - @_

- We've seen this one already
  - Also know as @ARG, but again, no one uses this

- ```
  mysub($foo, $bar, $baz);
  sub mysub {
      my ($name, $color, $car) = @_;
      ...
  }
  ```
  ```

The Command Line Arguments - @ARGV

- We've also seen this one
- # ./myprogram.pl file1 file2 file3

```
for $file (@ARGV) {
    open FILE, "<$file";
    ...
}
```


ARGV and $ARGV

- We've even seen ARGV, but you didn't know it
- while (<>) is the same as while (<ARGV>)
- ARGV – the special filehandle that iterates over the command-line filenames in @ARGV. Almost never written – implicit <> used instead.
- $ARGV – Contains the name of the current file when reading from ARGV using <>

$. - Current Record Number

- $. holds the current record number (typically line number) when you are reading from a file using a filehandle
- while (<FH>) {

```
    # Look ma, cheap line numbering!
    printf "%-6s $_", $.;
}
```


$/ - Record Separator

- $/ is the input record separator. Newline by default. Consulted by <FH> and chomp()
- Undefining $/ is useful to slurp file into a scalar
  - undef $/;
    $entirefile = <FH>;
  - {

```
    local $/;                # safer way to do the same thing
    $entirefile = <FH>;      # I'll tell you why, but then forget
}                            # why :-)
```

$/ Continued

- Setting it to null string will consider blank line to be record separator
- $/ = ''; # That's two single quotes there
  while $chunk (<FH>) {
  # $chunk now holds everything that was in FH
  # up to the next blank line
  }
- Usually we just leave $/ alone

$”

- $” specifies the string to put between each element when you interpolate an array inside of double quotes
- @array = qw /red green blue/;
  print “@array\n”; # “red green blue”
  $” = '|';
  print “@array\n”; # “red|green|blue”

$a and $b – Sort Variables

- We've seen these in the homework
- $a and $b hold the two terms that sort() is currently sorting. You can use them to define custom behavior.
- # get the keys ordered based on their value
  @keys = sort { $hash{$a} <=> $hash{$b} }
  keys %hash;

The Environment

- The %ENV hash holds the environment of the shell in which Perl is running. The contents will vary from system to system.
- $0 is the name of the current perl program file
- $$ is the process id of the current perl program
- $] is the version of the perl interpreter being used
- STDIN/STDOUT/STDERR – We know these

Special Variables Summary

- This was just a taste of Perl's special variables – there are over 100.
- Most of them are really obscure
- More importantly, many of them can really mess up a program
- Especially if you are new to Perl, most not mentioned here are best left alone, unless you happen to like headaches :-)