Working with files

To perform any input or output in Perl you need to associate the name of a file with a name in the program. The name in the program is called a filehandle and the mapping is done with the open function.

Filehandles

- Filehandles are similar to other variables except they are not prefixed by any special character
- A filehandle is a name for a file, device, pipe or socket
  - File – a stream of characters
  - Device – screen, printer, serial ports, …
  - Pipe – a connection from one program to another
  - Socket – used in network communications. Sockets are endpoints of communications

- There are three filehandles Perl automatically opens when the program begins execution
  - STDIN – connect to the user's keyboard
  - STDOUT – connected to the user's display
    - print and printf send output to STDOUT unless a filehandle is explicitly named
  - STDERR – generally connected to the user's display
    - Indicates a location where the program will output error messages

Open function

To use a file it must first be opened and associated with a filehandle. This is done with the open function

The open function requires two arguments

- The first argument is the filehandle
- The second argument is the filename
  - The filename can include path information
  - If no path is specific Perl looks in the same directory as the program
  - The filename and path may be operating system dependent

```
open (HANDLE, “filename”) or die (“Error message: $!”);
```

If the open statement succeeds the HANDLE can be used to access the specified file

If the open statement fails the die function is executed

die and warn

- die function
  - `die (list)`
  - The die function is used to write a message to STDERR (screen by default) and exit the program
  - $! is a special variable that holds the latest system-error message
  - If a newline is not attached to the last element in the die function the current script filename, line number and input line number (if any) are appended to the message, along with a final newline
- warn function
  - `warn (list)`
  - The warn function is used to write a message to STDERR but does not exit the program
  - The warn function will attach the same information to the list if is does not a newline is not attached to the last element in the list

Filename and filemodes

The file may be opened in one of several modes by appending prefixes to the filename

To read an existing file, open it for reading

```
“filename” or “<filename”
```

To create a file or to delete an existing file before writing data open the file for writing

```
“>filename”
```

To append data to an existing file, or to create it new if it doesn’t exist

```
“>>filename”
```

To open a file for reading and writing and keeping the existing file

```
“+<filename”
```

To open a file for reading and writing and discarding an existing file

```
“+>filename”
```

Reading the file

Data is received sequentially from the file, normally starting from the beginning of the file and reads all the data consecutively.

Line input operator (diamond operator) is used to read data

```
$input = <HANDLE>;
```

Accepts input from the file

Always returns characters up to and including the next newline character

Reads data into the special variable $_ if no variable is assigned to <HANDLE>

The HANDLE inside the line input operator can be a reference to a scalar variable that contains the name of the filehandle

Just like reading from the keyboard, the <> operator reads in data until it reaches a character that matches the value in $/

Writing to a file

To write information to a file use the print statement

```
print HANDLE infoToPrint;
```

This is the same print statement used to send information to the screen

- The HANDLE is optional and output is sent to the default output destination if not specified
  - STDOUT is the default output destination
- If HANDLE is specified there is no comma after the HANDLE

printf function also takes an optional HANDLE argument before the information to print

Close function

Every file should be closed as soon as it is known that the program will not reference the file again.

- Can help prevent file corruption
- Can free up resources for other users or programs

If you don’t close files explicitly Perl will close them when the program finishes execution

Calling the open statement for a file without closing the file will close the file before reopening it

```
close (HANDLE);
```

or

```
close (HANDLE) or die (“Error message: $!”);
```

eof function

The eof function detects when the program is at the end of the file referenced by HANDLE.

```
eof (HANDLE);  #uses HANDLE file
eof  #uses last file read
```

Returns 1 if either the last read operation moved the file pointer to the end of the file (the most previous file input read the last of the file) or if the file was not opened

select function

The select function lets the program change the default output destination

```
select (HANDLE);
```

Print statements without a filehandle will send output to HANDLE. STDOUT will have to be used to send output to the screen.

```
print “Goes to the HANDLE file\n”;
print STDOUT “Goes to the screen\n”;
```

`select (STDOUT)` will return standard output to the screen

Special variables

$! will hold the value of the latest system error

$. contains the line number of the current filehandle. Closing the file resets this variable

ARGV

- Perl stores the command line arguments (excluding any Perl options and the command name itself) in the special array @ARGV
  - $#ARGV contains the index of the last item of the array
  - $0 holds the name of the Perl script

File operators

File operators return information about a file. Placing the file operator in front of the file will return the type of data in parenthesis.

- The file is not a HANDLE, it is a filename with or without a path
- Some operators will only be useful in UNIX/LINUX operating systems

-r	file is readable (boolean)
-w	file is writable (boolean)
-x	file is executable (boolean)
-f	file is a plain file (boolean)
-d	file is a directory (boolean)
-e	file exists (boolean)
-B	file is a binary file (boolean)
-T	file is a text file (boolean)
-M	age of file in days (integer)
-A	age since last accessed (integer)

Pipes

The pipe mechanism takes the output of one program and feed that data to another program or command

The | symbol tells the system to send/receive data using a “pipe”.

- The data from the other process must be read from/written into a filehandle.

If your program should receive data from a process/program the | symbol must be placed after the process/program name

```
open (HANDLE, “process | “);
```

To receive data from the process read the data from HANDLE

If your program should send data to a process/program the | system must be place before the process/program name

```
open (HANDLE, “| process“);
```

To send data to the process write the data to HANDLE

Data filehandle

The DATA filehandle is generally used for debugging programs. It allows for easy and consistent data entry into programs. It works in conjunction with the keyword __END__.

If a Perl program contains only __END__ left-adjusted on a line then the Perl script will not process any Perl code beneath that line. Data can be placed on the lines following that line. Reading from the DATA filehandle will read from the data listed after __END__

```
while ($lineOfData = <DATA>)
{
   print $lineOfData;
}
__END__
Sample Data – Line 1
Line 2
Line 3
```