

# Getting Tools to Fit the Job. Using Components, Pipes, Redirection, etc

Jim Jackson <jj@franjam.org.uk>

May 12, 2003

This document can be found at <http://www.comp.leeds.ac.uk/jj/linux/cli2.html>

---

## Contents

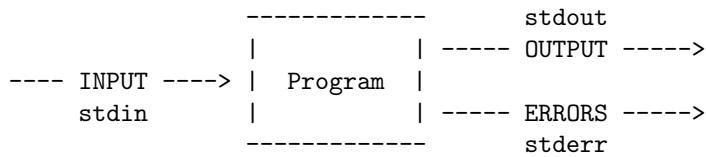
<b>1 Program Input and Output</b>	<b>3</b>
<b>2 Redirection</b>	<b>4</b>
2.1 Redirecting STDOUT . . . . .	4
2.2 Redirecting STDIN . . . . .	4
2.3 Redirecting STDERR . . . . .	4
2.4 Redirecting ALL output to same place . . . . .	4
<b>3 Joining programs together - PIPES</b>	<b>5</b>
3.1 Joining STDOUT of one command with STDIN of a second command . . . . .	5
3.2 Joining STDERR and STDOUT of one command to STDIN of a second command . . . . .	5
<b>4 Some of the main CLI utilities</b>	<b>6</b>
<b>5 Some of the main CLI "filters"</b>	<b>7</b>
<b>6 Example pipelines</b>	<b>8</b>

In this, any reference to a SHELL is assumed to be either the Bourne shell or one of its descendants, e.g. BASH.

---

# 1 Program Input and Output

## The Conventional Model



Each program when executed has 3 files preset for it:

- One input file, **STDIN**
- and 2 separate outputs files, **STDOUT** and **STDERR**.

**STDIN** is usually the keyboard

**STDOUT** is usually the console, terminal or shell window

**STDERR** is usually the console, terminal or shell window

**Remember:** In Unix/Linux “everything is a file” - even the keyboard/screen

The program can of course completely ignore these and get its inputs, and set its outputs to anything it wants. But most of the standard Unix/Linux command line tools follow a convention in the use of these preset files.

- If there is no alternative input file(s) specified, then the program takes its input from **STDIN**
- It directs its normal output to **STDOUT**
- It directs any error messages etc to **STDERR**

OK Why is this useful? ...

---

## 2 Redirection

... Because when executing programs, the **SHELL** can change what “file” is connected to **STDIN**, **STDOUT** and **STDERR**.

### 2.1 Redirecting STDOUT

Use the ‘>’ or ‘>>’ symbols to change where the normal output goes. ‘>>’ causes the output to be appended to any previous contents of the file.

```
e.g. % ls -l /bin > /tmp/bin_list
      % ls -l /usr/bin >> /tmp/bin_list
```

### 2.2 Redirecting STDIN

Use the ‘<’ symbol to change where the normal input comes from

```
e.g. % wc -l < /tmp/etc_list
```

mmmmm..... compare with “wc -l /tmp/etc\_list”, it’s only a small change. A less forced e.g.

```
% /usr/sbin/sendmail -t -oem -oi < /tmp/mailfile
```

### 2.3 Redirecting STDERR

Use ‘2>’ to change where the error output goes, **2** because **STDERR** is file **2**. To append use ‘2>>’

```
e.g. % ls -l Z* 2> /dev/null
```

### 2.4 Redirecting ALL output to same place

Use the construct ‘> **outputfile** 2>&1’. That is, redirect **STDOUT** to **file** and redirect file 2 (**STDERR**) to file 1 (&1)

```
e.g.      % ( ./configure ; make ) > /tmp/make.output 2>&1
or better % ( ./configure && make ) > /tmp/make.output 2>&1
```

---

## 3 Joining programs together - PIPES

The **shell** can use a **PIPE** (see man pipe) to feed the output of one program into the input of another.

### 3.1 Joining STDOUT of one command with STDIN of a second command

Use the '|' symbol between the 2 commands.

e.g. `% ls -l /etc | less`

### 3.2 Joining STDERR and STDOUT of one command to STDIN of a second command

Use the construct '`2>&1 |`'

e.g. `% ( ./configure ; make) 2>&1 | less`  
or better `% ( ./configure && make) 2>&1 | less`

Because programs can be so joined together, the UNIX convention of making utilities do essentially one thing, but do it well, has developed. More complex actions can be produced by joining commands together with pipes.

---

## 4 Some of the main CLI utilities

### some programs that generate useful info

- **ls** list directory contents
- **ps** report process status
- **file** determine file type
- **du** estimate file space usage
- **dmesg** print or control the kernel ring buffer

### miscellany

- **mail** send and receive mail
  - **sendmail** an electronic mail transport agent  
even if sendmail is not installed, you might find that a program called sendmail is there!
  - **man** an interface to the on-line reference manuals  
see also **info** , if you must.
-

## 5 Some of the main CLI "filters"

These all work on files, which can be specified on the command line. If no filenames are specified then they take their input from **STDIN**

- **cat** concatenate files and print on STDOUT
  - **tac** , **rev** see man pages
  - **less** less is more, but better
  - **head** output the first part of files
  - **tail** output the last part of files
  - **tee** read from STDIN and write to STDOUT and files
  
  - **wc** print the number of bytes, words, and lines in files
  - **grep** print lines matching a pattern
  - **sort** sort lines of text files
  - **uniq** remove duplicate lines from a sorted file
  - **cut** remove sections from each line of files
  - **sed** a Stream Editor
-

## 6 Example pipelines

show the mozilla processes running...

```
% ps wuax | grep mozill
```

show the number of files in the `_Docs_` directory...

```
% ls Docs | wc -l
```

mail yourself a list of files in your archive directory...

```
% ls -l archive | mail -s "List of Current Archive files" jj
```

browse the kernel messages from the `syslog` file...

```
% grep kernel /var/log/messages | less
```

watch for `tcpwrapper` connect messages in `syslog` file and save seperately...

```
% tail -f /var/log/syslog | grep "connect from"
```

---

HomePage <http://www.comp.leeds.ac.uk/jj>