Shell's file descriptors and I/O redirection

File descriptors (fds)

In *nix-like operating systems **everything** considered to be a **file**. There are **special files** and **regular files**. **File descriptor** is a **link**, that points to a **file**. Each program reads it's input from **link** number **0** (**file descriptor 0**), prints it's output to **link** number **1** (**file descriptor 1**), prints it's error output to **link** number **2** (**file descriptor 2**). The program **does not care** about where the **link** is pointing, it just sends the stream to a **link**. **Links** can be changed on a **once-only basis** (**df 1>** *file*): for the **df** command, **link** number **1** (**file descriptor 1**) will point to *file* instead of **stdout**. **Links** can be changed **permanently** with **exec** command (**exec 3<** *file*): **link** number **3** (**file descriptor 3**) will point to a *file*, the *file* will be opened in **read mode**. **Links** (**file descriptors**) can be closed by linking them with **&-** (**exec 3<&-**). **Link** itself (**file descriptor**) can be copied to another **link** (**file descriptor**), **exec 4>&1** will make a copy of **link** number **1** (**file descriptor 1**) in **link** number **4** (**file descriptor 4**) with **file mode** being **write**. Later on, **exec 1>&4** can be used to restore **link** number **1** (**file descriptor 1**) to it's original state.

Upon invocation, each shell, subshell, subsubshell gets **3** of it's own **unique** links:

1) – A **link** to **stdin**, available on **file descriptor 0**

2) – A **link** to **stdout**, available on **file descriptor 1**

3) – A **link** to **stderr**, available on **file descriptor 2**

A stream can be redirected to **fd**, specific to this particular shell/subshell

| – Pipes inherit **fds**; to prevent that from happening, **fd** needs to be closed (**4>&-**)

2>&1 – Links **fd2** with **fd1** on a once-only basis (**find** / -name '*.h' **2>&1**)

exec 4>&1 – Links **fd4** with **fd1** permanently (**fd4** will contain exact copy of **fd1** link)

Interactive shell requires **fd0**, **fd1**, **fd2** to be open at all times, and they must contain unique links to shell's **stdin**, **stdout**, **stderr**; thus **exec 2>&1** will fail in interactive shell

Non-interactive shell allows **fd0**, **fd1**, **fd2** to be linked with other files:

exec 3<&0 – Saves **fd0** (**unique link**, pointing to shell's **stdin**) in **fd3**

exec 0< f1 – **fd0** is now pointing at file **f1**, which is open for reading

read -- var – reads **line** from file **f1** instead of **stdin**, stores value in variable **var**

exec 0<&3 – Restores **fd0** to it's original state.

fd0 and **fd1** – Numbers on the left can be omitted for these 2 **fds**: ls > *file* is same as ls **1>** *file*

fd5 – **bash** shell uses **fd5** for its internal purposes. **Never** use **fd5** in bash scripts!

Redirecting stream from subsubshell into subshell

```
#!/bin/ksh

# Preparations

if [ ! -d restricted ]; then mkdir restricted; chmod 000 restricted; fi

# Redirection

var=$( ( ls -R 2>&1 1>&4 | sed 's/denied/denied for you/' 1>&2 ) 4>&1 )
```

var – A variable in the parent shell

subshell – **$(4>&1)**

subsubshell – **(ls** -R **2>&1 1>&4 | sed** 's/denied/denied for you/' **1>&2)**

All these redirections are applied only to **one** command they're used together with.

Shell **evaluates** the **whole command line** before executing it:

1) – In subshell **4>&1** links **fd4** to whatever is in **fd1**, here it's subshell's unique **stdout**

2) – In subsubshell **2>&1** links subsubshell's **fd2** to it's own unique **stdout**

3) – In subsubshell **1>&4** links subsubshell's **fd1** to **fd4**

4) – In subsubshell **1>&2** links subsubshell's **fd1** to it's own unique **stderr** on **fd2**

Error stream from **ls** -R goes into **stdout** (because now, for **ls** -R command and left part of pipe, **fd2** is linked to subsubshell's unique **stdout**); then it goes through pipe **|** to be received by **sed**, then **sed**'s output is redirected into subsubshell's unique **stderr** on **fd2**. As a result, error stream is modified and sent back to **stderr**.

Normal stream from **ls** -R goes into **fd4**, and is received in subshell, where **4>&1** links **fd4** with subshell's unique **stdout**: now subsubshell's **fd1** is linked with subshell's unique **stdout**. Value, returned by subshell is then assigned to variable var in the parent shell.

fd4 can be closed to prevent it's stream from going through pipe with **4>&-**, it isn't necessary here:

```
var=$( ( ls -R 2>&1 1>&4 4>&- | sed 's/denied/denied for you/' 1>&2 4>&- ) 4>&1 )
```

Summary – parent shell, subshell, subsubshell, subsubsubshell, ... all have their own unique links to **stdin**, **stdout**, **stderr**. All steams can be redirected into each subshell, subsubshell, subsubsubshell, ...

Redirecting two streams from subsubshell into one fd in subshell

#!/bin/ksh

set -o noglob

Preparations

print 'ONE\nTWO\nTHREE' **1>|** *f1*

 # Redirection

set -A **ar $(** ***(print*** *"***$(0<** *f1* **)"** **|**

while ***read*** *v; do* ***print*** *"$v"* ***1>&4****;* ***print*** *'*****'* ***1>&6****; done* ***)*** **4>&1 6>&1)**

ar – An array in the parent shell

subshell – **$(4>&1 6>&1)**

subsubshell – **(print** "" **|** while **read** v; do **print** "$v" **1>&4**; **print** '*****' **1>&6**; done **)**

subsubsubshell – **$(0<** *f1* **)**

1) – In subshell **4>&1** links **fd4** to **fd1**, **6>&1** links **fd6** to **fd1** (subshell's **stdout**)

2) – In subsubshell **1>&4** links subsubshell's **fd1** to **fd4** for the 1st **print** command

3) – In subsubshell **1>&6** links subsubshell's **fd1** to **fd6** for the 2nd **print** command

4) – In subsubsubshell **0<** *f1* links **fd0** with file *f1;* file contents returned to subsubshell

Both **print** commands in subsubshell have their **file descriptors** number **1** linked to the same unique **stdout** of subshell, where both streams are being sent. The array in parent shell will be:

${ar[0]} – ONE

${ar[1]} – *****

${ar[2]} – TWO

${ar[3]} – *****

${ar[4]} – THREE

${ar[5]} – *****

Summary – this example shows how **many** streams can be redirected to one **single file descriptor**. Steams from different shells, subshells, subsubshells, subsubsubshells, ..., all can be sent to one **single file descriptor**.

Receiving return status from the first element in the pipeline

#!/bin/ksh

Preparations

if [**!** **-d** *restricted*]; then **mkdir** *restricted*; **chmod** 000 *restricted*; fi

Redirection

exec **6>&1**

var=**$(** ***(*** **(** **ls** -R **2>&1** **1>&6**; **print** $? **1>&4** **)** **|** ***sed*** *'s/denied/denied for you/'* ***1>&2*** ***)*** **4>&1** **)**

exec **6>&1** – Links **fd6** with **fd1** in parent shell (unique **link** to parent shell's **stdout**)

var – A variable in the parent shell

subshell – **$(** **4>&1** **)**

subsubshell – **(** **|** **sed** 's/denied/denied for you/' **1>&2** **)**

subsubsubshell – **(** **ls** -R **2>&1** **1>&6**; **print** $? **1>&4** **)**

1) – In subshell **4>&1** links **fd4** to **fd1** (subshell's **stdout**)

2) – In subsubshell **1>&2** links subsubshell's **fd1** to it's own unique **stderr** on **fd2**

3) – In subsubsubshell **2>&1** links subsubshell's **fd2** to it's own unique **stdout**

4) – In subsubsubshell **1>&6** links subsubshell's **fd1** to **fd6** (for 1st command)

5) – In subsubsubshell **1>&4** links subsubshell's **fd1** to **fd4** (for 2nd command)

Error stream from **ls** -R goes into **stdout** (because now, for **ls** -R command, **fd2** is linked to subsubsubshell's unique **stdout** with **2>&1**); normal stream from **ls** -R goes into **fd6** (**1>&6**) and is received in the parent shell in it's **stdout** (because of **exec** **6>&1**) to be displayed on screen. **print** sends return status ($?) of **ls** -R command into **fd4** (**1>&4**), it's then received in subshell (**4>&1**) and then returned as a value to be assigned to variable var in the parent shell.

In subsubshell value, returned by subsubsubshell (from **ls -R**) is received by **sed** through pipe |, then **sed**'s output is redirected into subsubshell's unique **stderr** on **fd2**. As a result, error stream is modified and sent back to **stderr**.

Summary – a value from **ls -R** command in the subsubsubshell is sent to the parent shell's **stdout**; a value containing return status of **ls -R** command is sent to subshell's stdout and returned by subshell to be assigned to a variable var in parent shell. Error stream from **ls -R** command in the subsubsubshell is returned by subsubsubshell as a result into subsubshell, sent into **sed** through pipe, modified by **sed** and sent back into the error stream. Similarly, return status of **any part of the pipeline** can be retrieved. In parent shell **exec** **6>&1** can be replaced with **exec** **6>** *myFile* to send the normal output of **ls -R** command into a regular file.

Copyright (c) 2021 Ivan Bityutskiy

Permission to use, copy, modify, and distribute this documentation for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE DOCUMENTATION IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
WITH REGARD TO THIS DOCUMENTATION INCLUDING ALL IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS DOCUMENTATION.