Operating Systems
Redirecting File Descriptors

Distributed Systems Group
University of Minho

1 Objectives

Become familiar with and use the system calls relating to the duplication of file descriptors and their application in redirection, more specifically the *standard* descriptors of input, output and error.

2 System Calls

```
#include <unistd.h>     /* system calls: essential defs and decls */

int dup(int fd);
int dup2(int fd1, int fd2);
```

3 Exercises

1. Write a program that redirects the descriptor associated with its *standard input* to the file `/etc/passwd`, and the *standard output* and *error* to `output.txt` and `error.txt`, respectively. Immediately before the program finishes, it should print the message "finished" on the terminal.

2. Modify the previous program so that, after performing the redirections, a new process is created that performs the read and write operations. Look at the contents of the files. Notice that the child process is *"born"* with the same file descriptor associations as the parent process.

3. Modify the initial program again so that the command `wc` is executed, without arguments, after the input and output descriptors have been redirected. Note that, once again, the associations – and redirections – of file descriptors are preserved by the primitive `exec()`.

4. Write a program that runs the command `wc` in a child process. The parent process must send to the child through an anonymous pipe a sequence of text lines entered by the user in its *standard input*. Use the redirection technique to associate the *standard input* of the child process with the read descriptor of the anonymous pipe created by the parent. Recall the need to close the write descriptor(s) in the pipe in order to verify the *end of file* situation.

5. Write a program that emulates the operation of the command interpreter in the chained execution of `ls /etc | wc -l`.

6. Write a program that emulates the operation of the command interpreter in the chained execution of `grep -v ^# /etc/passwd | cut -f7 -d: | uniq | wc -l`.

4 Additional exercises

1. Add to the command interpreter proposed in guide 3 the possibility of redirecting the inputs, outputs and errors of the commands it executes. Consider the redirection operators `<`, `>`, `>>`, `2>` and `2>>`.

2. Add to the command interpreter proposed in the previous guides the possibility of chaining the inputs and outputs of programs to be executed through anonymous pipes (`|` operator).