

**Umeå University**  
Department of Computing Science

**System Level Programming 7.5 p**  
**5DV088**

**Minimal Shell - MISH**

Submitted 2016-10-07  
Author: Lorenz Gerber (dv15lgr@cs.umu.se lozger03@student.umu.se)  
Instructor: Mikael Ränner / Filip Åberg / Jonathan Westin / Mattias Åsander

## Contents

1	Problem Description	1
2	Compilation and Invocation	1
3	Detailed Usage Description	1
4	Algorithm Description	1
5	System Description	2
6	Known Limits	2
7	Results	2
8	Discussion	2
9	Testing	2
	References	2

## 1 Problem Description

The aim of this laboration was to develop a minimal unix shell. The shell had to implement input and output stream redirection, stream piping between commands, and two internal commands, 'cd' and 'echo'. The main techniques to be used for implementing the shell were 'pipes', 'forking' and execution of programs by the 'exec' function family. Further, the interrupt signal (SIGINT) had to be reassigned to a function that allows to stop all child processes and then returns control back to the shell prompt.

## 2 Compilation and Invocation

The present code uses safe signaling by 'sigaction' which requires on linux to compile with std=gnu11 instead of ANSI C standard. On OSX, the code built also when using std=c11, hence the gcc included in linux seems to have a more stringent interpretation of 'c11' standard ('sigaction' is POSIX, not ANSI C). A makefile is provided that takes care of building and linking the source by invoking 'make all'. A 'make clean' script is also included.

## 3 Detailed Usage Description

'mish' does not take any command line argument. The shell can be quit by sending EOF to the prompt (Ctrl-D). SIGINT (Ctrl-C) is ignored in the prompt main loop and assigned to a custom function during execution of external commands. The custom implementation stops all started child processes and returns back to the command prompt.

## 4 Algorithm Description

```
loop over all but the last commands
  create a pipe
  spawn a child process
```

Child process code:

```
  if requested
    setup redirect
  else
    close pipe read_end
  if 'in' != 0
    dup2 'in' to STDIN
  dup2 pipe write_end to STDOUT
  close pipe write_end
  run command
```

Parent process code:

```
  if 'in' != 0, close 'in'
  set 'in' to pipe read_end
  close pipe write_end
```

2(2)

spawn a child process for last command

Child process code:

```
if there is only one command and this command has
    to redirect input from a file
    then redirect in_file file descriptor to STDIN
```

```
if redirect of output to a file is requested
    redirect out_file file descriptor to STDOUT
```

```
redirect pipe read_end to STDIN
```

```
clean up
```

```
run command
```

Parent process code:

```
wait for child processes to report finished
cleaning up
```

## **5 System Description**

## **6 Known Limits**

The program does not compile with the `std=c11` or `ansi` flag set on Linux (tested on the standard gcc version shipped with ubuntu 16.04). However it does compile on OSX (clang-800.0.38).

## **7 Results**

## **8 Discussion**

## **9 Testing**

### **9.1 Testing for closing of file descriptors**

### **9.2 Testing for correct finishing of child processes**