

SigCopy

Deadline: October 10th, 2014

1 Instructions

You must implement two programs `sigsend` and `sigrecv`, working as follows:

- `sigsend` takes a single command-line argument that identifies a file to send. It must print its process ID and start waiting for a client. When a client appears, it must transfer the file's contents to the client and terminate.
- `sigrecv` takes a single command-line argument that identifies a `sigsend` process (via its PID). It must interact with that process to receive the remote file. The received bytes must be printed to the standard output.

Example session:

```
# in shell 1
$ echo hello>test.txt
$ ./sigsend test.txt
12314
```

```
# in shell 2
$ ./sigrecv 12314
hello
```

- Your programs must use `SIGUSR1` and `SIGUSR2` for the data transfer. Tip: `sigaction/siginfo_t`.
- You may use any standard C function (either from ISO C 1999/2011 or POSIX); however your code may not use any descriptor-based channel (file, pipe, socket, etc.) other than the standard output, nor use the SysV or POSIX “msg”, “mq”, “shm” or “sem” IPC facilities.
- You may not use `system` or any other mechanism that invokes an external program.

2 Grading

- 6 points if your programs can transfer text files successfully.
- +2 points if your programs can transfer any binary file successfully.
- +1 point if your programs work even when the size of the file is not known in advance (eg. reading from `/dev/stdin`).

- +1 point if your programs are robust to the spurious insertion of extra signals during the transfer.
-

3 Copyright and licensing

Copyright © 2014, Raphael 'kena' Poss. Permission is granted to distribute, reuse and modify this document and other documents for the Systems Programming course by the same author according to the terms of the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/>.