ECE357, Computer Operating Systems – Problem Set #1, Problem 3

Nikola Janjušević

September 20, 2018

minicat.c

```
/*
Nikola Janjusevic
Operating Systems Assignment 1
minicat.c -b -o infile
concatenates files with optional buffer size argument -b, and output file
argument -o. Files to cat are specified at the infile position.
Special character '-' (a single hyphen) may be used to allow for input from
the standard input.
*/
// argument parsing adapted from gnu.org example of getopt
# include <ctype.h>
# include <stdio.h>
# include <stdlib.h>
# include <unistd.h>
# include <fcntl.h>
# include <string.h>
# include <errno.h>

# define BUFFSIZE 4069

int main (int argc, char **argv)
{
  int i, n, fd_in;
  int buff_size = BUFFSIZE;
  int fd_out = STDOUT_FILENO;
  char *outfile = "";
  char *buf;
  char c;
  // argument parsing
  while( (c = getopt(argc, argv, "b:o:")) != -1)
  {
    switch(c)
    {
      case 'b':
        if( (buff_size = atoi(optarg)) < 1)
        {
          fprintf(stderr,"buffer size error: invalid input\n");
          exit(-1);
        }
        break;
      case 'o':
        outfile = optarg;
        if( (fd_out = open(outfile, O_CREAT|O_WRONLY|O_TRUNC, \
          S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH)) < 0)
        {
          fprintf(stderr,"outfile \"%s\" open error: %s\n", \
            outfile,strerror(errno));
          exit(-1);
        }
        break;
    }
  }
```

```
// buffer allocation
buf = malloc(buff_size);
// if NO INFILES are given
if(optind == argc)
{
  while( (n = read(STDIN_FILENO, buf, buff_size)) > 0)
  {
    if(write(fd_out, buf, n) != n)
    {
      fprintf(stderr,"outfile \"%s\" write error: %s\n", \
        outfile,strerror(errno));
      exit(-1);
    }
  }
  if(n < 0)
  {
    fprintf(stderr, "infile [stdin] read error: %s\n", strerror(errno));
    exit(-1);
  }
}
else
{ // if INFILES ARE GIVEN
  for(i = optind; i < argc; i++)
  {
    if(strncmp(argv[i], "-", 1) == 0) // hyphen infile specifies stdin
      fd_in = STDIN_FILENO;
    else if( (fd_in = open(argv[i], O_RDONLY)) < 0)
    {
      fprintf(stderr,"infile \"%s\" open error: %s\n", \
        argv[i],strerror(errno));
      exit(-1);
    }
    if(strcmp(argv[i], outfile) == 0)
    {
      fprintf(stderr,"error: infile \"%s\" is outfile\n", \
        outfile);
      exit(-1);
    }
    while( (n = read(fd_in, buf, buff_size)) > 0)
    {
      if(write(fd_out, buf, n) != n)
      {
        fprintf(stderr,"outfile \"%s\" write error: %s\n", \
          outfile,strerror(errno));
        exit(-1);
      }
    }
    if(n < 0)
    {
      fprintf(stderr,"infile \"%s\" read error: %s\n", \
        argv[i],strerror(errno));
      exit(-1);
    }
    if(fd_in != STDIN_FILENO && close(fd_in) < 0) // close the infile
    {
      fprintf(stderr, "infile \"%s\" close error: %s\n", \
        argv[i], strerror(errno));
      exit(-1);
```

```
      }
    }
  }
  if(fd_out != STDOUT_FILENO && close(fd_out) < 0) // close the outfile
  {
    fprintf(stderr, "outfile \"%s\" close error: %s\n", \
      outfile, strerror(errno));
    exit(-1);
  }
  free(buf);
  return 0;
}
```

Demonstration of `minicat`

```
$ ./minicat.out -o ex1.txt
this is example file number 1.
$ ./minicat.out -o ex2.txt
this is example file 2. neato.
$ ./minicat.out -b16 -o out.txt - ex1.txt - ex2.txt -
hello, this is the start of out.txt
this is the text between ex1.txt and ex2.txt
now, this is the end of out.txt. goodbye!
$ ./minicat.out out.txt
hello, this is the start of out.txt
this is example file number 1.
this is the text between ex1.txt and ex2.txt
this is example file 2. neato.
now, this is the end of out.txt. goodbye!
```

Demonstration of Error Reports

open error

```
$ ./minicat.out -o mcout ur1 nonexistentfile
infile "nonexistentfile" open error: No such file or directory
```

read error

```
$ ./minicat.out -o mcout ur1 ./bkup
infile "./bkup" read error: Is a directory
```

write error

A flash-drive was removed while `minicat` was attempting to write to it.

```
$ ./minicat.out -b256 -o /media/nikopj/nikola64/usbtest ur1 ur2 &
[2] 10561
$ outfile "/media/nikopj/nikola64/usbtest" write error: Input/output error
```

Effect of Buffer-Size on Runtime

Tested by concatenating two 2^{29} byte or $\sim$ 536 MB files copied from `/dev/urandom`. Smaller byte sizes were not tested because

```
$ time ./minicat.out -b512 -o mcout ur1 ur2

real 0m4.048s
user 0m0.442s
sys 0m3.308s

$ time ./minicat.out -b4096 -o mcout ur1 ur2

real 0m1.365s
user 0m0.065s
sys 0m1.021s

$ time ./minicat.out -b16777216 -o mcout ur1 ur2

real 0m1.348s
user 0m0.001s
sys 0m1.071s
```

From reading 4096 bytes at a time to reading 16 MB, the real time elapsed doesn't decrease substantially and the system time actually increases.

Output Comparison to `cat`

```
$ time cat ur1 ur2 > catout

real 0m0.561s
user 0m0.008s
sys 0m0.553s

$ sum catout mcout
56122 1048576 catout
56122 1048576 mcout
```

`minicat.out` is shown to have the same output as `cat`, even with binary files.

Question to ponder

How can you specify an input file which is literally a single hyphen, and not have it be confused with a command-line flag or the special symbol for standard input?

The way to do this is to refer to the hyphen file by its path name, ie. `./-`. This works under the current framework of the `minicat` program with no tweaks!

example:

```
$ ./minicat.out -o ./-
this is the hyphen file.
$ ./minicat.out -o out.txt ex1.txt ./-
$ ./minicat.out out.txt
this is example file number 1.
this is the hyphen file.
```