Homework H2: Chapter 10 Homework Problems

Solutions

Problem 10.6
What is the output of the following program?

```
#include "csapp.h"
int main()
 {
  int fd1, fd2;
  fd1 = Open("foo.txt", O_RDONLY, 0);
  fd2 = Open("bar.txt", O_RDONLY, 0);
  Close(fd2);
  fd2 = Open("baz.txt", O_RDONLY, 0);
  printf("fd2 = %d\n", fd2);
  exit(0);
 }
```

Problem 10.6 Solution:
On entry, descriptors 0-2 are already open. The open function always returns the lowest possible descriptor, so the first two calls to open return descriptors 3 and 4. The call to the close function frees up descriptor 4, so the final call to open returns descriptor 4, and thus the output of the program is “fd2 = 4".

Problem 10.8
Write a version of the statcheck program in Figure 10.10, called fstatcheck, that takes a descriptor number on the command line rather than a file name.

Problem 10.8 Solution:

```
#include "csapp.h"

int main (int argc, char **argv)
 {
  struct stat stat;
  char *type, *readok;
  int size;

  if (argc != 2)
   {
    fprintf(stderr, "usage: %s <fd>\n", argv[0]);
    exit(0);
   }
  Fstat(atoi(argv[1]), &stat);
  if (S_ISREG(stat.st_mode)) /* Determine file type */
  type = "regular";
  else if (S_ISDIR(stat.st_mode))
  type = "directory";
  else if (S_ISCHR(stat.st_mode))
  type = "character device";
  else
  type = "other";

  if ((stat.st_mode & S_IRUSR)) /* Check read access */
  readok = "yes";
  else
  readok = "no";

  size = stat.st_size; /* check size */
```

```
  printf("type: %s, read: %s, size=%d\n",
  type, readok, size);

  exit(0);
 }
```

Problem 10.10
Modify the cpfile program in Figure 10.4 so that it takes an optional command line argument infile. If infile is given, then copy infile to standard output; otherwise, copy standard input to standard output as before. The twist is that your solution must use the original copy loop (lines 9-11) for both cases. You are only allowed to insert code, and you are not allowed to change any of the existing code.

Problem 10.10 Solution:

```
#include "csapp.h"

int main(int argc, char **argv)
 {
  int n;
  rio_t rio;
  char buf[MAXLINE];

  if (argc == 2)
   {
    int fd;
    if ((fd = Open(argv[1], O_RDONLY, 0)) < 0)
     {
      fprintf(stderr, "Couldn't read %s\n", argv[1]);
      exit(1);
     }
    Dup2(fd, STDIN_FILENO);
    Close(fd);
   }

  Rio_readinitb(&rio, STDIN_FILENO);
  while((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0)
  Rio_writen(STDOUT_FILENO, buf, n);
  exit(0);
 }
```