File IO

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

File Flag Testing (1)

```
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

int main(void) {
  int fd1, fdx;
  int i = 42;

  puts ("* using <<<          >>>");

  fd1 = open("t1.txt",          );
  fdx = open("tx.txt",          );

  if (fd1 == -1) perror ("fail to open t1.txt");
  else puts("t1.txt is opened successfully");

  if (fdx == -1) perror ("fail to open tx.txt");
  else puts("tx.txt is opened successfully");

  if ((fd1 == -1) || (fdx == -1)) exit(1);

  close(fd1);
  close(fdx);
}
```

In m1.c

O_WRONLY

In m2.c

O_WRONLY | O_CREAT

In m3.c

O_WRONLY | O_TRUNC

In m4.c

O_WRONLY | O_EXCL

In m5.c

O_WRONLY | O_EXCL | O_CREAT

In m6.c

O_WRONLY | O_TRUNC | O_CREAT

File Flag Testing (2)

In run.bash

```
#!/bin/bash

gcc -Wall -o m1 m1.c
gcc -Wall -o m2 m2.c
gcc -Wall -o m3 m3.c
gcc -Wall -o m4 m4.c
gcc -Wall -o m5 m5.c
gcc -Wall -o m6 m6.c


commands=("m1" "m2" "m3" "m4" "m5" "m6")

for cmd in ${commands[@]}; do
  echo "===================================="
  echo "Executing $cmd"

  echo -n "this is the content of t1.txt" > t1.txt
  if [ -f "tx.txt" ]; then
    rm -f tx.txt
  fi

  echo "* before executing"
  ls -l t*.txt
  echo "------------------------------------"
  ./${cmd}
  echo "------------------------------------"
  echo "* after executing"
  ls -l t*.txt
  echo " "

done
```

In m1.c

O_WRONLY

In m2.c

O_WRONLY | O_CREAT

In m3.c

O_WRONLY | O_TRUNC

In m4.c

O_WRONLY | O_EXCL

In m5.c

O_WRONLY | O_EXCL | O_CREAT

In m6.c

O_WRONLY | O_TRUNC | O_CREAT

t1.txt is always an existing file
tx.txt is an non-existing file

File Flag Testing (3)

```
====================================
Executing m1
* before executing
-rw-r--r-- 1 young young 29 Oct 11 19:16 t1.txt
------------------------------------
* using <<< O_WRONLY >>>
t1.txt is opened successfully
fail to open tx.txt: No such file or directory
------------------------------------
* after executing
-rw-r--r-- 1 young young 29 Oct 11 19:16 t1.txt

====================================
Executing m2
* before executing
-rw-r--r-- 1 young young 29 Oct 11 19:16 t1.txt
------------------------------------
* using <<< O_WRONLY | O_CREAT >>>
t1.txt is opened successfully
tx.txt is opened successfully
------------------------------------
* after executing
-rw-r--r-- 1 young young 29 Oct 11 19:16 t1.txt
---------- 1 young young  0 Oct 11 19:16 tx.txt
```

```
====================================
Executing m3
* before executing
-rw-r--r-- 1 young young 29 Oct 11 19:16 t1.txt
------------------------------------
* using <<< O_WRONLY | O_TRUNC >>>
t1.txt is opened successfully
fail to open tx.txt: No such file or directory
------------------------------------
* after executing
-rw-r--r-- 1 young young 0 Oct 11 19:16 t1.txt

====================================
Executing m4
* before executing
-rw-r--r-- 1 young young 29 Oct 11 19:16 t1.txt
------------------------------------
* using <<< O_WRONLY | O_EXCL >>>
t1.txt is opened successfully
fail to open tx.txt: No such file or directory
------------------------------------
* after executing
-rw-r--r-- 1 young young 29 Oct 11 19:16 t1.txt
```

File Flag Testing (4)

```
====================================
Executing m5
* before executing
-rw-r--r-- 1 young young 29 Oct 11 19:16 t1.txt
------------------------------------
* using <<< O_WRONLY | O_EXCL | O_CREAT >>>
fail to open t1.txt: File exists
tx.txt is opened successfully
------------------------------------
* after executing
-rw-r--r-- 1 young young 29 Oct 11 19:16 t1.txt
---------- 1 young young  0 Oct 11 19:16 tx.txt

====================================
Executing m6
* before executing
-rw-r--r-- 1 young young 29 Oct 11 19:16 t1.txt
------------------------------------
* using <<< O_WRONLY | O_TRUNC | O_CREAT >>>
t1.txt is opened successfully
tx.txt is opened successfully
------------------------------------
* after executing
-rw-r--r-- 1 young young 0 Oct 11 19:16 t1.txt
---------- 1 young young 0 Oct 11 19:16 tx.txt
```

Write binary data

```
// in wr_type.c
#include <stdio.h>
#include <string.h>

int main() {
    FILE *fp1, *fp2, *fp3, *fp4;

    char  cbuf = 0x12;       // 1-byte buffer
    short sbuf = 0x1234;     // 2-byte buffer
    int   ibuf = 0x12345678; // 4-byte buffer
    float fbuf ;             // 4-byte buffer

    // copy the bit pattern without type casting
    memcpy(&fbuf, &ibuf, sizeof(int));

    fp1 = fopen("c.bin", "wb");
    fp2 = fopen("s.bin", "wb");
    fp3 = fopen("i.bin", "wb");
    fp4 = fopen("f.bin", "wb");

    if ((fp1 == NULL) || (fp2 == NULL) ||
        (fp3 == NULL) || (fp4 == NULL)) {
        perror("Failed to open file");
        return 1;
    }

    fwrite(&cbuf, sizeof(cbuf), 1, fp1);
    fwrite(&sbuf, sizeof(sbuf), 1, fp2);
    fwrite(&ibuf, sizeof(ibuf), 1, fp3);
    fwrite(&fbuf, sizeof(fbuf), 1, fp4);

    fclose(fp1); fclose(fp2); fclose(fp3); fclose(fp4);

    return 0;
}
```

Shell Script

```
#!/bin/bash

gcc -o wr_type wr_type.c
gcc -o rd_type rd_type.c


./wr_type

xxd c.bin
xxd s.bin
xxd i.bin
xxd f.bin

./rd_type
```

```
00000000: 12                                       .
00000000: 3412                                     4.
00000000: 7856 3412                                xV4.
00000000: 7856 3412                                xV4.
cbuf = 0x12 18
sbuf = 0x1234 4660
ibuf = 0x12345678 305419896
fbuf = 0x12345678 5.690457e-28
```

Read binary data

```
#include <stdio.h>
#include <string.h>

int main() {
    FILE *fp1, *fp2, *fp3, *fp4;

    char  cbuf ; // 1-byte buffer
    short sbuf ; // 2-byte buffer
    int   ibuf ; // 4-byte buffer
    float fbuf ; // 4-byte buffer
    int   ival ;

    fp1 = fopen("c.bin", "rb");
    fp2 = fopen("s.bin", "rb");
    fp3 = fopen("i.bin", "rb");
    fp4 = fopen("f.bin", "rb");

    if ((fp1 == NULL) || (fp2 == NULL) ||
        (fp3 == NULL) || (fp4 == NULL)) {
        perror("Failed to open file");
        return 1;
    }

    fread(&cbuf, sizeof(cbuf), 1, fp1);
    fread(&sbuf, sizeof(sbuf), 1, fp2);
    fread(&ibuf, sizeof(ibuf), 1, fp3);
    fread(&fbuf, sizeof(fbuf), 1, fp4);

    // copy the bit pattern without type casting
    memcpy(&ival, &fbuf, sizeof(int));

    printf("cbuf = 0x%02X %d \n", cbuf, cbuf);
    printf("sbuf = 0x%04X %d \n", sbuf, sbuf);
    printf("ibuf = 0x%08X %d \n", ibuf, ibuf);
    printf("fbuf = 0x%08X %e \n", ival, fbuf);

    fclose(fp1); fclose(fp2); fclose(fp3); fclose(fp4);

    return 0;
}
```

Read binary data

```
#include <stdio.h>
#include <string.h>

int main() {
    FILE *fp1, *fp2, *fp3, *fp4;

    char  cbuf ; // 1-byte buffer
    short sbuf ; // 2-byte buffer
    int   ibuf ; // 4-byte buffer
    float fbuf ; // 4-byte buffer
    int   ival ;

    fp1 = fopen("c.bin", "rb");
    fp2 = fopen("s.bin", "rb");
    fp3 = fopen("i.bin", "rb");
    fp4 = fopen("f.bin", "rb");

    if ((fp1 == NULL) || (fp2 == NULL) ||
        (fp3 == NULL) || (fp4 == NULL)) {
        perror("Failed to open file");
        return 1;
    }

    fread(&cbuf, sizeof(cbuf), 1, fp1);
    fread(&sbuf, sizeof(sbuf), 1, fp2);
    fread(&ibuf, sizeof(ibuf), 1, fp3);
    fread(&fbuf, sizeof(fbuf), 1, fp4);

    // copy the bit pattern without type casting
    memcpy(&ival, &fbuf, sizeof(int));

    printf("cbuf = 0x%02X %d \n", cbuf, cbuf);
    printf("sbuf = 0x%04X %d \n", sbuf, sbuf);
    printf("ibuf = 0x%08X %d \n", ibuf, ibuf);
    printf("fbuf = 0x%08X %e \n", ival, fbuf);

    fclose(fp1); fclose(fp2); fclose(fp3); fclose(fp4);

    return 0;
}
```

Shell Script

```
#!/bin/bash

gcc -o wr_numbers wr_numbers.c
gcc -o rd_bigbuf  rd_bigbuf.c
gcc -o rd_fseek   rd_fseek.c
gcc -o rd_4bytes  rd_4bytes.c


./wr_numbers

xxd numbers.bin

./rd_bigbuf
./rd_fseek
./rd_4bytes
```

```
00000000: 0001 0203 0405 0607 0809 0a0b 0c0d 0e0f  ................
00000010: 1011 1213 1415 1617 1819 1a1b 1c1d 1e1f  ................
00000020: 2021 2223 2425 2627 2829 2a2b 2c2d 2e2f   !"#$%&'()*+,-./
00000030: 3031 3233 3435 3637 3839 3a3b 3c3d 3e3f  0123456789:;<=>?
00000040: 4041 4243 4445 4647 4849 4a4b 4c4d 4e4f  @ABCDEFGHIJKLMNO
00000050: 5051 5253 5455 5657 5859 5a5b 5c5d 5e5f  PQRSTUVWXYZ[\]^_
00000060: 6061 6263 6465 6667 6869 6a6b 6c6d 6e6f  `abcdefghijklmno
00000070: 7071 7273 7475 7677 7879 7a7b 7c7d 7e7f  pqrstuvwxyz{|}~.
00000080: 8081 8283 8485 8687 8889 8a8b 8c8d 8e8f  ................
00000090: 9091 9293 9495 9697 9899 9a9b 9c9d 9e9f  ................
000000a0: a0a1 a2a3 a4a5 a6a7 a8a9 aaab acad aeaf  ................
000000b0: b0b1 b2b3 b4b5 b6b7 b8b9 babb bcbd bebf  ................
000000c0: c0c1 c2c3 c4c5 c6c7 c8c9 cacb cccd cecf  ................
000000d0: d0d1 d2d3 d4d5 d6d7 d8d9 dadb dcdd dedf  ................
000000e0: e0e1 e2e3 e4e5 e6e7 e8e9 eaeb eced eeef  ................
000000f0: f0f1 f2f3 f4f5 f6f7 f8f9 fafb fcfd feff  ................
```

```
* executing rd_bigbuf ...

a[ 0] = 0x03020100 50462976
a[ 1] = 0x07060504 117835012

a[62] = 0xFBFAF9F8 -67438088
a[63] = 0xFFFEFDFC -66052


* executing rd_fseek ...

a[ 0] = 0x03020100 50462976
a[ 1] = 0x07060504 117835012

a[62] = 0xFBFAF9F8 -67438088
a[63] = 0xFFFEFDFC -66052


* executing rd_fseek ...

a[ 0] = 0x03020100 50462976
a[ 1] = 0x07060504 117835012
```

wr_number.c

```
#include <stdio.h>

int main() {
    FILE *file;
    int i;
    unsigned char buf; // 1-byte buffer

    file = fopen("numbers.bin", "wb");

    if (file == NULL) {
        perror("Failed to open file");
        return 1;
    }

    for (i = 0; i <= 0xFF; ++i) {
        buf = (unsigned char)i;
        fwrite(&buf, sizeof(buf), 1, file);
    }

    fclose(file);
    return 0;
}
```

rd_bigbuf.c

```
#include <stdio.h>
#include <stdint.h>

int main() {
    FILE *file;
    int buf[64];  // 256 byte buffer

    int i, j;
    int a[64];  // read 64 integers from 256 byte stream

    file = fopen("numbers.bin", "rb");

    if (file == NULL) {
        perror("Failed to open file");
        return 1;
    }

    if (fread(buf, sizeof(int), 64, file) != 64) {
        perror("Failed to read file");
        fclose(file);
        return 1;
    }

    fclose(file);

    for (int i = 0; i < 64; ++i) {
      a[i] = buf[i];
    }

    // Print each integer in hexadecimal

    puts("\n\n* executing rd_bigbuf ...\n");
    for (int i = 0; i < 64; ++i) {
        printf("a[%2d] = 0x%08X %d \n", i, a[i], a[i]);
    }

    return 0;
}
```

rd_fseek.c

```
#include <stdio.h>
#include <stdint.h>

int main() {
    FILE *file;

    int buf;
    int i;
    int a[256 / 4];  // 64 integers

    file  = fopen("numbers.bin", "rb");

    if (file == NULL) {
        perror("Failed to open file");
        return 1;
    }

    for (i = 0; i < 64; ++i) {
        // Move file pointer to the correct position
        if (fseek(file, i * 4, SEEK_SET) != 0) {
            perror("fseek failed");
            fclose(file);
            return 1;
        }

        // Read 4 bytes from the current position
        if (fread(&buf, sizeof(buf), 1, file) != 1) {
            perror("Failed to read 4 byte integer");
            fclose(file);
            return 1;
        }

        a[i] = buf;
    }

    fclose(file);

    // Print the integers in hexadecimal

    puts("\n\n* executing rd_fseek ...\n");
    for (int i = 0; i < 64; ++i) {
        printf("a[%2d] = 0x%08X %d\n", i, a[i], a[i]);
    }

    return 0;
}
```

rd_4bytes.c

```
#include <stdio.h>
#include <stdint.h>

int main() {
    FILE *file;

    int a[256 / 4];  // 64 integers
    unsigned char buf[4];

    file  = fopen("number.bin", "rb");

    if (file == NULL) {
        perror("Failed to open file");
        return 1;
    }

    for (int i = 0; i < 64; ++i) {
        if (fread(buf, sizeof(unsigned char), 4, file) != 4) {
            perror("Failed to read 4 bytes");
            fclose(file);
            return 1;
        }

        // Combine 4 bytes into an int (little-endian)
        a[i] = (buf[0] << 0)
             | (buf[1] << 8)
             | (buf[2] << 16)
             | (buf[3] << 24);
    }

    fclose(file);

    // Print each integer
    for (int i = 0; i < 64; ++i) {
        printf("a[%2d] = 0x%08X %d\n", i, a[i], a[i]);
    }

    return 0;
}
```

Write Characters and Read Integers using buf

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    char *cbuf, *cptr; // 256 char buffer cbuf and a pointer cptr
    int  *ibuf, *iptr, *jptr; // 64 int buffer ibuf and pointers iptr, jptr
    int i;

    cbuf = (char *) malloc(256 * sizeof(char));  // 256 bytes
    ibuf = (int  *) malloc( 64 * sizeof(int));   // 256 bytes

    if (cbuf == NULL || ibuf == NULL) {
        printf("Memory allocation failed!\n");
        return 1;
    }

    // writes 256 one-byte char numbers to cbuf
    cptr = cbuf;  // make cptr point to the 1st element of cbuf
    for (i = 0; i <= 0xFF; ++i) {
        memcpy(cptr++, &i, sizeof(char)); // cptr moves by 1 byte
    }


    // copies 64 4-byte int numbers to ibuf from cbuf
    iptr = ibuf;  // make iptr point to the 1st element of ibuf
    jptr = (int *) cbuf; // and jptr point to the 1st element of cbuf
    for (i = 0; i < 64; ++i) {
        memcpy(iptr++, jptr++, sizeof(int)); // iptr & jptr moves by 4 bytes
        printf("ibuf[%2d] = 0x%08X %d \n", i, ibuf[i], ibuf[i]);
    }


    return 0;
}
```

References

[1] http://minix1.woodhull.com/current/2.0.4/
[2]