明治大学理工学部情報科学科

ソフトウェア実習 レポート

3章 ファイル入出力

提出日 2025 年 10 月 13 日

3 年 15組 91番 157R257501

SEO BEOMGYO

はじめに

本課題の目的は、標準Cライブラリのファイル入出力関数（fopen, fclose, getc, putc, fscanf等）を用いて、ファイルサイズの取得、ファイルのコピー・比較、データの読み書きなど基本的なファイル操作を実装し、エラー処理を含めた堅牢なプログラム作成技術を習得することである。対象課題はファイルサイズ取得、ファイルコピー、複数ファイルへの同時コピー、catコマンド実装、文字コード出現頻度集計、ファイル比較、整数データの読み取りと集計、カンマ区切りデータの解析の8課題であり、いずれもEOF検出とエラー処理を適切に行う必要がある。

課題1：sample.txtのサイズ取得

```c
// pointer
char* strchr1(const char* s, int c) {
    while (*s) {
        if (*s == c) return (char*)s;
        s++;
    }
    return NULL;
}

// index
char* strchr2(const char* s, int c) {
    int i = 0;
    while (s[i] != '\0') {
        if (s[i] == c) return (char*)(s + i);
        i++;
    }
    return NULL;
}
```

目的

sample.txtという名前のファイルを開き、getc関数でEOFまで読み取ることでファイルサイズ（バイト数）を取得し、ls -lコマンドの結果と一致することを確認する。

実装要点

- fopen関数で”r”モードによりファイルを開き、失敗時はperrorでエラーメッセージを出力する。
- whileループでgetc関数を呼び出し、EOFに達するまでカウンタcountを加算する。
- fclose関数でファイルを閉じ、最終的なバイト数を出力する。

テストと実行結果

- 入力：sample.txtに”Hello¥n”（6バイト）を格納
- 出力：”sample.txt：6 byte”
- ls -l sample.txtの結果と一致することを確認した。

課題2：Unixのcpコマンドの基本機能実装

```
int ch;
// EOF に達するまで
while ((ch = getc(in)) != EOF) {
    if (putc(ch, out) == EOF) {
        fprintf(stderr, "書き込みエラー : %s: %s\n", dst, strerror(errno));
        fclose(in);
        fclose(out);
        return 4;
    }
}
```

目的

2つのファイル名をコマンドライン引数で受け取り、最初のファイルの内容を2番目のファイルにコピーするプログラムを作成する。

実装要点

- argc != 3の場合、引数不足エラーとしてリターンコード1を返す。
- 入力ファイルを"rb"モード、出力ファイルを"wb"モードで開き、エラー時にはstrerror(errno)で詳細を出力する。
- getc/putcでEOFまで1バイトずつコピーし、ferrorでストリームエラーを検査する。
- fcloseの返り値も検査し、失敗時はエラーメッセージを出力する。

テストと実行結果

- 入力：./2 sample.txt copy.txt
- 出力：copy.txtがsample.txtと同一内容で作成される

課題3：2つのファイルへの同時コピー

```
int ch;
// EOFまで1バイトずつ読み取り、2つの出力へ同一データを書き込む
while ((ch = getc(in)) != EOF) {
    if (putc(ch, out1) == EOF || putc(ch, out2) == EOF) {
        fprintf(stderr, "書き込みエラー : %s or %s\n", out1_name, out2_name);
        fclose(in);
        fclose(out1);
        fclose(out2);
        return 4;
    }
}
```

目的

sample1.txtの内容をsample2.txtとsample3.txtの2つのファイルに同時にコピーするプログラムを作成する。

実装要点

- 入力ファイルを”rb”モード、2つの出力ファイルをそれぞれ”wb”モードで開く。
- getcで読み取った1バイトをputcで両方の出力ファイルに書き込む。
- いずれかのputcが失敗した場合、即座にエラー処理を行い全ストリームを閉じる。
- ferrorによる読み取りエラー検査とfcloseの返り値検査を全ストリームに実施する。

テストと実行結果

- 入力：./3（sample1.txtを読み取り）
- 出力：sample2.txt, sample3.txtがsample1.txtと同一内容で作成される
- cmpコマンドで3つのファイルが一致することを確認した。

課題4：任意個のファイルを順に標準出力

```c
int cat_file(const char *name) {
    FILE *fp = fopen(name, "rb");
    if (!fp) {
        fprintf(stderr, "読み取りエラー : %s: %s\n", name, strerror(errno));
        return 1;
    }

    int ch;
    while ((ch = getc(fp)) != EOF) {
        if (putchar(ch) == EOF) {
            fprintf(stderr, "書き込みエラー : %s\n", strerror(errno));
            fclose(fp);
            return 2;
        }
    }

    if (ferror(fp)) { // 読み取り途中のエラー
        fprintf(stderr, "読み取りエラー : %s: %s\n", name, strerror(errno));
        fclose(fp);
        return 3;
    }

    if (fclose(fp) == EOF) { // クローズ時のエラー
        fprintf(stderr, "クローズエラー : %s: %s\n", name, strerror(errno));
        return 4;
    }

    return 0;
}
```

目的

コマンドライン引数で指定された複数のファイルを順次標準出力に連結して表示するcatコマンド相当の機能を実装する。

実装要点

- argc < 2の場合、引数不足エラーとしてリターンコード1を返す。

- cat_file関数で各ファイルを”rb”モードで開き、getcで読み取った内容をputcharで標準出力に書き出す。
- ferrorによるストリームエラー検査とfcloseの返り値検査を実施する。
- 1つでもエラーが発生した時点で処理を終了する。

テストと実行結果

- 入力: ./4 sample1.txt sample2.txt
- 出力: sample1.txtと sample2.txtの内容が順次標準出力に表示される
- Unixのcat sample1.txt sample2.txtと同じ挙動を確認した。

課題5: ファイル名1つを引数として文字出現回数表示

```
printf("code  char  count\n");
printf("-----------------------\n");
for (int i = 0; i < 256; i++) {
    if (counts[i] == 0) continue;
    if (is_printable(i)) {
        printf("%3d   %c     %llu\n", i, i, counts[i]);
    } else {
        printf("%3d         %llu\n", i, counts[i]);
    }
}
```

目的

ファイル名をコマンドライン引数で受け取り、0～255の各バイト値の出現回数を集計して一覧表示するプログラムを作成する。

実装要点

- unsigned long long countsの配列を全てゼロで初期化する。
- getcでファイル全体を走査し、各バイト値のカウンタを加算する。
- is_printable関数でASCII 32～126の範囲を判定し、印字可能文字は文字も併記する。
- 出現回数が0でないコードのみを表示する。

テストと実行結果

- 入力: ./5 sample.txt
- 出力:

text

code char count

----

 32   1

```
 72  H    1
 87  W    1
100  d    1
101  e    1
108  l    3
111  o    2
114  r    1
```

課題6：ファイル名2つを引数としてファイル比較

```c
while (1) {
    ca = getc(fa);
    cb = getc(fb);
    pos++;

    // printf("ca = %d, cb =%d\n", ca, cb);

    if (ca == EOF || cb == EOF) {
        if (ca == EOF && cb == EOF) {
            // 完全一致
            printf("(1) identical\n");
            fclose(fa); fclose(fb);
            return 0;
        } else if (ca == EOF) {
            // Aの方が先にEOF → Aが短い
            printf("(3) EOF on A at byte %llu (A shorter)\n", pos);
            fclose(fa); fclose(fb);
            return 10;
        } else {
            // Bの方が先にEOF → Bが短い
            printf("(4) EOF on B at byte %llu (B shorter)\n", pos);
            fclose(fa); fclose(fb);
            return 11;
        }
    }

    if (ca != cb) {
        printf("(2) differ: byte %llu, A=0x%02X, B=0x%02X\n", pos, (unsigned char)ca, (unsigned char)cb);
        fclose(fa); fclose(fb);
        return 12;
    }
}
```

目的

2つのファイル名をコマンドライン引数で受け取り、バイト単位で比較して、(1)完全一致、(2)差分位置とバイト値、(3)Aが短い、(4)Bが短い、のいずれかを判定して出力する。

実装要点

- 両ファイルを"rb"モードで開き、getcで逐次読み取りながら位置カウンタposを加算する。
- ca == EOF || cb == EOFの場合、両方EOFなら完全一致、片方のみEOFなら短い方を

報告する。

- ca != cbの場合、差分位置と両者のバイト値を16進表記で出力し、リターンコード12を返す。
- Unixのcmpコマンドと同等の機能を実現する。

テストと実行結果

- テストケース(1): ./6 fileA.txt fileA.txt → 出力: "(1) identical"
- テストケース(2): ./6 short.txt long.txt → 出力: "(2) differ: byte 5, A=0x41, B=0x42"
- テストケース(3): ./6 short.txt longer.txt → 出力: "(3) EOF on A at byte 10 (A shorter)"
- テストケース(4): ./6 longer.txt short.txt → 出力: "(4) EOF on B at byte 10 (B shorter)"

課題7: integers.txtから整数を読み取り合計と個数を出力

```
// EOF に達するまで
while (fscanf(fp, "%d", &x) == 1) {
    printf("%d\n", x);
    sum = sum + x;
    count++;
}
```

目的

integers.txtファイルに書き込まれた10進整数形式の複数の整数をfscanfで読み取り、各整数を表示した後、合計と個数を出力するプログラムを作成する。

実装要点

- fopen("integers.txt", "r")でテキストモードで開く。
- while (fscanf(fp, "%d", &x) == 1)により正常に読み取れた整数に対してループ処理を行う。
- sumとcountを累積し、ferrorによるストリームエラー検査とfcloseの返り値検査を実施する。
- fscanfの返り値が1である間のみループを継続することで、数値以外の文字やEOFに正しく対応する。

テストと実行結果

- 入力ファイルintegers.txtの内容:

text

10

-5

32

7

- 出力:

text

10

-5

32

7

sum=44

count=4

課題8: ”10,31”形式のカンマ区切り整数ペアを読み取り

```
char* strpbrk1(const char* s, const char* accept) {
    for (; *s != '\0'; s++) {
        for (const char *a = accept; *a != '\0'; a++) {
            if (*s == *a) return (char*)s;
        }
    }
    return NULL;
}
```

目的

コマンドライン引数で指定されたファイルに記述された”10,31”形式のカンマ区切り整数ペアをfscanfで読み取り、各ペアとその和を出力するプログラムを作成する。

実装要点

- argc != 2の場合、引数不足エラーとしてリターンコード1を返す。
- fopen(path, ”r”)でテキストモードで開く。
- ループ内でfscanfの返り値rを検査し、r == 2なら正常なペア読み取り、r == EOFなら終了、それ以外は形式エラーとする。
- fscanf(fp, ”%d,%d”, &a, &b)により、カンマを区切り文字として明示的に指定する。

テストと実行結果

- 入力ファイルの内容:

```
text
10,31
5,15
-3,8
```

- 出力:

```
text
10
31
10 + 31 = 41
5
15
5 + 15 = 20
-3
8
-3 + 8 = 5
```

考察

全課題を通じて、ファイル入出力の基本操作であるfopen, fclose, getc, putc, fscanfの使用法、EOF検出、エラー処理（fopen失敗、ferrorによるストリームエラー、fclose失敗）を網羅的に実装した。