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

4章 簡易版 cat コマンドの作成

提出日 2025 年 10 月 20 日

3 年 15組 91番 157R257501

SEO BEOMGYO

はじめに

本課題の目的は、UNIXのcatコマンドの簡易版を実装し、複数ファイルの順次出力、標準入力からの入力、オプション解析（-n, -h, -t, -b）の処理を通じて、コマンドライン引数の解析、ファイル入出力、行番号の管理、エラー処理を含む実用的なCプログラムの作成技術を習得することである。

課題1～4：catコマンドの基本機能と拡張オプションの実装

```c
typedef struct {
    bool use_lineno;   // -n 指定の有無
    long h_start;      // 開始行
    long t_end;        // 終了行
} Opts;

static void print_usage(const char *prog) {
    fprintf(stderr,
        "Usage: %s [-n] [-h<num>] [-t<num>] [file ...]\n"
        "  -n       各行の先頭に通し行番号を付与\n"
        "  -h<num> 出力開始行を指定 (num>=1)\n"
        "  -t<num> 出力終了行を指定 (num>=1, -h と併用可)\n", prog);
}

// 10進整数の妥当性検査つき変換
static bool parse_pos_long(const char *s, long *out) {
    if (!s || !*s) return false;
    char *end = NULL;
    long v = strtol(s, &end, 10);
    if (*end != '\0') return false;
    if (v < 1 || v == LONG_MAX || v == LONG_MIN) return false;
    *out = v;
    return true;
}
```

```c
// 1ストリームを処理
static int process_stream(FILE *fp, const Opts *opt, long *p_line, int *p_done) {
    int c;
    int bol = 1;          // 行頭フラグ
    long line = *p_line;  // 現在の通し行番号

    while ((c = fgetc(fp)) != EOF) {
        if (opt->t_end > 0 && line > opt->t_end) { *p_done = 1; break; }

        if (bol) {
            // 行頭に来たタイミングで行番号の出力可否を判断
            int in_range = (line >= opt->h_start) && (opt->t_end <= 0 || line <= opt->t_end);
            if (in_range && opt->use_lineno) {
                if (printf("%ld ", line) < 0) return -1;
            }
            bol = 0;
        }

        // 範囲内なら1文字ずつ出力、範囲外なら読み捨て
        if ((line >= opt->h_start) && (opt->t_end <= 0 || line <= opt->t_end)) {
            if (fputc(c, stdout) == EOF) return -1;
        }

        if (c == '\n') { line++; bol = 1; }
    }

    if (ferror(fp)) return -1;
    *p_line = line;
    return 0;
}

static int is_option(const char *s) {
    return s && s[0] == '-' && s[1] != '\0';
}
```

目的

課題1～4では、catコマンドの基本仕様（複数ファイルの連結出力、標準入力対応）に加え、-nオプション（行番号付与）、-hオプション（出力開始行指定）、-tオプション（出力終了行指定）の3種類のオプションを単一プログラムとして実装し、オプションの自由な組み合わせに対応する。

実装要点

基本機能

コマンドライン引数を解析し、is_option関数でハイフンで始まる文字列をオプション、それ以外をファイル名として判別する。

ファイル名が一つもない場合（argc == 1またはfile_count == 0）、標準入力（stdin）から読み取り処理を行う。

各ファイルを”rb”モードでfopenし、失敗時にはstrerror(errno)でエラーメッセージをstde

rrに出力して次のファイルへ継続する。

オプション解析の実装

Opts構造体を定義し、use_lineno（-nフラグ）、h_start（開始行）、t_end（終了行）を管理する。

strcmpで"-n"を判定し、strncmpで"-h"や"-t"の接頭辞を検出する。

parse_pos_long関数でstrtol関数を用いて"-h3"や"-t12"のような数値部分を抽出し、妥当性検査（1以上、LONG_MAX/LONG_MIN以外、終端が'¥0'）を行う。

未知のオプションに対してはprint_usage関数でエラーメッセージとヘルプを表示してリターンコード2で終了する。

-nオプションの実装

process_stream関数内で行頭フラグ（bol）を管理し、行頭に来たタイミングで範囲内かつuse_lineno == trueの場合に行番号を出力する。

改行（'¥n'）を検出した時に行番号を加算し、ポインタ（*p_line）により複数ファイルにわたって通し行番号を維持する。

-h/-tオプションの実装

論理的整合性チェックとして、h_start > t_endの場合はエラーメッセージを出力して終了する。

process_stream関数内で現在行番号（line）が範囲内（line >= h_start かつ（t_end <= 0 または line <= t_end））の場合のみfputcで文字を出力し、範囲外は読み捨てる。

t_endを超えた時点で*p_doneフラグを1にして全ファイルの処理を早期終了する。

ストリーム処理とエラー処理

process_stream関数で1ストリームを処理し、fgetcで1文字ずつ読み取りfputcで標準出力に書き出す。

ferror(fp)でストリームエラーを検査し、エラー時は-1を返す。

main関数のループで各ファイルを順次処理し、エラー時もexit_codeを1に設定して次のファイルへ継続する。

テストと実行結果

基本機能のテスト

入力: % ./mycat -n test.txt test.txt

出力:

```
text
1 This
2     is
3         test.
4 This
5     is
6         test.
```

入力: % ./mycat -h2 -n -t5 test.txt test.txt

出力:

```
text
2     is
3         test.
4 This
5     is
```

課題5（オプショナル）：-bオプションの実装

```c
typedef struct {
    bool use_n;
    bool use_b;
    long h_start;
    long t_end;     // -1 = 無制限
} Opts;

static void print_usage(const char *prog){
    fprintf(stderr,
      "Usage: %s [-n|-b] [-h<num>] [-t<num>] [file ...]\n"
      "  -n       すべての行に通し番号\n"
      "  -b       空行を除いて番号（範囲内では h から連番）\n"
      "  -h<num> 開始行, -t<num> 終了行\n", prog);
}

static int parse_pos_long(const char *s, long *out){
    if(!s || !*s) return 0;
    char *e=NULL; long v=strtol(s,&e,10);
    if(*e!='\0' || v<1 || v==LONG_MAX || v==LONG_MIN) return 0;
    *out=v; return 1;
}
```

目的

-bオプションを追加し、全部で4種類とする。-bは-nと同様だが、空行には番号を付けずに出力する。-hや-tオプションとの併用時の仕様を定義し実装する。

実装要点

Opts構造体にuse_b（bool型）を追加し、-bオプションの有無を管理する。

-bオプションが指定された場合、use_bをtrueにし、-nオプションより優先して処理する。

物理行番号（phys）で-h/-tの範囲を判定し、論理番号（vln）は範囲内の非空行に対してのみh_startから開始して連番を付与する。

行頭（bol）で次の文字が改行（'¥n'）でない場合のみ行番号を出力し、vlnを加算する。空行（改行のみの行）は行番号を付けずにそのまま出力する。

vlnの初期化は範囲内に初めて入った非空行で行い、その時点でvln = h_startとする。

テストと実行結果

入力ファイルに空行が含まれる場合、-bオプション使用時は空行に行番号を付けない

-h/-tオプションとの併用時、範囲内の非空行にのみ行番号を付与する

論理番号は範囲内で-hで指定した値から開始する

考察

-h/-tオプションの範囲判定と-nオプションの行番号維持を両立させるため、ポインタによる状態共有と範囲外データの読み捨て処理を導入し、効率的かつ堅牢な実装となった。

parse_pos_long関数による数値の妥当性検査、h_start > t_endの論理的整合性チェック、未知のオプションに対するエラー処理により、実用的なコマンドラインツールとしての完成度を高めた。また、ファイルオープン失敗時に次のファイルへ継続する設計により、部分的なエラーでも可能な限り処理をした。