

大作业报告

（ 2018 / 19 学年 第 一 学期）

题 目：**对常见Linux命令及其实现的分析**

|  |  |
| --- | --- |
| **专 业** | **数字媒体技术** |
| **班 级 学 号** |  |
| **学 生 姓 名** |  |
| **指 导 教 师** | **王蔚** |
| **指 导 单 位** | **计算机学院、软件学院、网络空间安全学院 计算机软件教学中心** |
| **日 期** | **2019年1月10日** |

**对常见Linux命令及其实现的分析**

**一、命令概述**

Cmp命令用来对比两个文件的内容，以字节为单位。如果两个文件没有任何差异，命令不会回显任何信息。如果发现差异，默认会显示出第一个差异对应的列数编号。如果不指定任何文件路径或者给定的文件路径位“-”cmp命令会从标准输入设备读取数据用以对比。

Cmp [OPTION]... FILE1 [FILE2 [SKIP1 [SKIP2]]]

可选参数SKIP1和SKIP2 是用来指定掉过文件开头和结尾的字节数目，默认数值是0，跳过0个字节。

**二、用户接口分析**

1. 命令可以命令行方式执行，用户通过构建命令行命令调用程序。用户需要指定两个文件路径作为比较对象。还有许多可选的参数。最基础的用法举例：

Cmp file1 file2

以字节位单位比较两个文件的内容差异。

如我们创建两个文件file1，和file2，在这两个文件中分别键入abcdef和abcdfe。然后使用该命令对比两个文件。得到的回显结果如下：

file1 file2 differ: byte 4, line 1

回显表明在文件第四个字节的位置，第一行发现差异。如果是两个完全没有差别的文件，这条命令默认不会有回显。

**三、概要分析**

1．全局数据结构

/\* 存储本程序的程序名称 \*/

char const \*program\_name;

/\* 用以存储传入的两个文件的文件名 \*/

static char const \*file[2];

/\* 两个文件的文件描述 \*/

static int file\_desc[2];

/\* 用来读取文件而开辟的两个缓冲区 \*/

static char \*buffer[2];

/\* 最佳文件块大小 \*/

static size\_t buf\_size;

/\* 初始的每个文件忽略的头字节 \*/

static off\_t ignore\_initial;

/\* 输出格式:

type\_first\_diff:打印第一个差别字节所在的偏移地址和行数

type\_all\_diffs:打印十进制格式的偏移量以及八进制格式的所有有差别字节

type\_status:只返回一个信息用来标明两个文件是否存在差异

\*/

static enum {

type\_first\_diff, type\_all\_diffs, type\_status

} comparison\_type;

/\* 如果非零，则以cat -t 的格式排列打印字节的值\*/

static int opt\_print\_chars;

/\* 存储各个长参数名对应的短参数名 \*/

static struct option const long\_options[] = {

{"print-chars", 0, 0, 'c'},

{"ignore-initial", 1, 0, 'i'},

{"verbose", 0, 0, 'l'},

{"silent", 0, 0, 's'},

{"quiet", 0, 0, 's'},

{"version", 0, 0, 'v'},

{"help", 0, 0, 129},

{0, 0, 0, 0}

};

2. 功能模块说明

(1) main( )

函数原型：int main(int argc,char \*argv[])

头文件：无

参数：argc,argv[]

返回值：int

返回cmp()函数的返回结果

功能描述：

负责启动程序

负责调用其它功能模块以实现相应的功能

此函数需要调用的其它函数（调用关系）：

usage();

check\_stdout();

try\_help();

cmp();

函数定义文件：cmp.c

(2) cmp()

函数原型：int cmp()

头文件：无

参数：无

返回值：int

返回type\_first\_diff：打印第一个差别字节所在的偏移地址和行数

返回type\_all\_diffs：打印十进制格式的偏移量以及八进制格式的所有有差别字节

返回type\_status：只返回一个信息用来标明两个文件是否存在差异

功能描述：

负责程序的主体功能，按字节比较对比两个文件或者标准输入流的差异。

此函数需要调用的其它函数（调用关系）：

File\_psoition()

Block\_compare()

block\_compare\_and\_count()

block\_read()

函数定义文件：cmp.c

(3) printc()

函数原型：static void printc (int width,unsigned c)

头文件：无

参数：width,c

返回值：无

功能描述：

打印单个字符c，以及处理相关不可打印字符

此函数需要调用的其它函数（调用关系）：无

函数定义文件：cmp.c

(4) usage()

函数原型：static void usage()

头文件：无

参数：无

返回值：无

功能描述：

负责打印程序的help页面，即程序命令的使用方法

此函数需要调用的其它函数（调用关系）：无

函数定义文件：cmp.c

(5) try\_help()

函数原型：static void try\_help()

头文件：无

参数：无

返回值：无

功能描述：

负责打印程序的try help信息

此函数需要调用的其它函数（调用关系）：无

函数定义文件：cmp.c

(6) block\_compare ()

函数原型：static size\_t block\_compare (char const \*p0,char const \*p1)

头文件：无

参数：p0, p1

返回值：size\_t

返回第一个差异字节的偏移量

功能描述：

被cmp比较函数主体调用，比较两块内存里从p0位置到怕

位置直到出现不同。

此函数需要调用的其它函数（调用关系）：无

函数定义文件：cmp.c

(7) block\_compare\_and\_count ()

函数原型：static size\_t block\_compare\_and\_count (char const \*p0, char const \*p1,long \*count)

头文件：无

参数：p0,p1,count

返回值：size\_t

返回第一个差异字节的偏移地址，以及count中返回遇到的“\n”字符的个数

功能描述：

比较两块内存地址从p0到p1的位置知道出现不同字节。并且记录遇到的“\n”字符的个数。

此函数需要调用的其它函数（调用关系）：无

函数定义文件：cmp.c

(8) block\_read ()

函数原型：static size\_t block\_read (int fd,char \*buf,size\_t n chars)

头文件：无

参数：fd, buf, chars

返回值：size\_t

返回成功读入的字符的个数。

功能描述：

将文件的内容读入buf中。

此函数需要调用的其它函数（调用关系）：无

函数定义文件：cmp.c

(9) check\_stdout ()

函数原型：static void check\_stdout ()

头文件：无

参数：无

返回值：无

功能描述：

检测标准输出流是否出现错误，出现错误输出错误报告。

此函数需要调用的其它函数（调用关系）：无

函数定义文件：cmp.c

(10) file\_position ()

函数原型：static off\_t file\_position (int i)

头文件：无

参数：i

返回值：off\_t

返回一个偏移地址

功能描述：

用来跳过文件中需要跳过的起始字节

此函数需要调用的其它函数（调用关系）：无

函数定义文件：cmp.c

**四、详细分析**

1. 主函数接受命令行参数，格式化处理参数内容，根据内容进行下一步处理。如显示主要用法以及对不文件差异等。

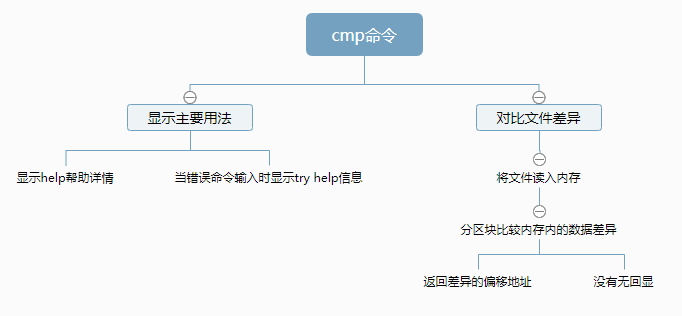


图1 cmp命令基本结构图

2.对比函数主体。首先依据参数判断是否有需要跳过的字节，并进行跳过操作。然后按区块从内存中读取文件内容，进行对比。

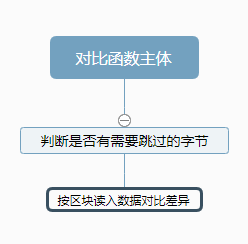


图2 cmp对比函数主体基本结构图

**五、源代码分析**

/\* main.c \*/

/\* cmp -- compare two files.

Copyright (C) 1990, 1991, 1992, 1993, 1994 Free Software Foundation, Inc.

This program is free software; you can redistribute it and/or modify

it under the terms of the GNU General Public License as published by

the Free Software Foundation; either version 2, or (at your option)

any later version.

This program is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

GNU General Public License for more details.

You should have received a copy of the GNU General Public License

along with this program; if not, write to the Free Software

Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA. \*/

/\* Written by Torbjorn Granlund and David MacKenzie. \*/

#include "system.h"

#include <stdio.h>

#include "getopt.h"

#include "cmpbuf.h"

extern char const version\_string[];

#if \_\_STDC\_\_ && defined (HAVE\_VPRINTF)

void error (int, int, char const \*, ...);

#else

void error ();

#endif

VOID \*xmalloc PARAMS((size\_t));

static int cmp PARAMS((void));

static off\_t file\_position PARAMS((int));

static size\_t block\_compare PARAMS((char const \*, char const \*));

static size\_t block\_compare\_and\_count PARAMS((char const \*, char const \*, long \*));

static size\_t block\_read PARAMS((int, char \*, size\_t));

static void printc PARAMS((int, unsigned));

static void try\_help PARAMS((char const \*));

static void check\_stdout PARAMS((void));

static void usage PARAMS((void));

/\* 存储本程序名称 \*/

char const \*program\_name;

/\* 存储需要对比的两个文件名 \*/

static char const \*file[2];

/\* 存储两个文件的内容描述 \*/

static int file\_desc[2];

/\* 文件读入缓冲 \*/

static char \*buffer[2];

/\* 最佳区块大小 \*/

static size\_t buf\_size;

/\* 需要忽略的起始字节数 \*/

static off\_t ignore\_initial;

/\* 输出格式 :

type\_first\_diff:打印差异字节偏移量以及行数

type\_all\_diffs:打印十进制的差异字节偏移量以及八进制的差异字节详情

type\_status:仅返回两个文件是否具有差异

\*/

static enum {

type\_first\_diff, type\_all\_diffs, type\_status

} comparison\_type;

#ifndef word

#define word int

#endif

/\*如果为零打印cat -t格式的差异字节内容. \*/

static int opt\_print\_chars;

/\* 存储长参数对应的短参数 \*/

static struct option const long\_options[] = {

{"print-chars", 0, 0, 'c'},

{"ignore-initial", 1, 0, 'i'},

{"verbose", 0, 0, 'l'},

{"silent", 0, 0, 's'},

{"quiet", 0, 0, 's'},

{"version", 0, 0, 'v'},

{"help", 0, 0, 129},

{0, 0, 0, 0}

};

/\* 打印try help信息 \*/

static void try\_help (char const \*reason;){

if (reason)

error (0, 0, "%s", reason);

error (2, 0, "Try `%s --help' for more information.", program\_name);

}

/\* 检查标准输出流是否出现错误，如果出现错误打印错误信息 \*/

static void check\_stdout () {

if (ferror (stdout))

error (2, 0, "write error");

else if (fclose (stdout) != 0)

error (2, errno, "write error");

}

/\* 打印usage 内容 \*/

static void usage () {

printf ("Usage: %s [OPTION]... FILE1 [FILE2]\n", program\_name);

printf ("%s", "\

-c --print-chars Output differing bytes as characters.\n\

-i N --ignore-initial=N Ignore differences in the first N bytes of input.\n\

-l --verbose Output offsets and codes of all differing bytes.\n\

-s --quiet --silent Output nothing; yield exit status only.\n\

-v --version Output version info.\n\

--help Output this help.\n");

printf ("If a FILE is `-' or missing, read standard input.\n");

}

/\* 主函数 \*/

int main (int argc,char \*argv[]){

int c, i, exit\_status;

struct stat stat\_buf[2];

initialize\_main (&argc, &argv); //格式化输入参数

program\_name = argv[0];

//按照命令行参数内容进行相应处理

while ((c = getopt\_long (argc, argv, "ci:lsv", long\_options, 0))

!= EOF)

switch (c) {

case 'c'://修改打印格式

opt\_print\_chars = 1;

break;

case 'i'://接受需要忽略的字节数

ignore\_initial = 0;

while (\*optarg) {

/\* Don't use `atol', because `off\_t' may be longer than `long'. \*/

unsigned digit = \*optarg++ - '0';

if (9 < digit)

try\_help ("non-digit in --ignore-initial value");

ignore\_initial = 10 \* ignore\_initial + digit;

}

break;

case 'l'://设置比较类型，全部不同均要比较

comparison\_type = type\_all\_diffs;

break;

case 's'://设置比较类型，仅回显是否有差异

comparison\_type = type\_status;

break;

case 'v'://打印版本信息

printf ("cmp - GNU diffutils version %s\n", version\_string);

exit (0);

case 129: //输出帮助信息

usage ();

check\_stdout ();

exit (0);

default: //打印try help信息

try\_help (0);

}

if (optind == argc)

try\_help ("missing operand");

file[0] = argv[optind++];

file[1] = optind < argc ? argv[optind++] : "-";

if (optind < argc)

try\_help ("extra operands");

for (i = 0; i < 2; i++) {

int i1 = i ^ (strcmp (file[1], "-") == 0);

if (i && filename\_cmp (file[0], file[1]) == 0)

exit (0);

file\_desc[i1] = (strcmp (file[i1], "-") == 0

? STDIN\_FILENO

: open (file[i1], O\_RDONLY, 0));

if (file\_desc[i1] < 0 || fstat (file\_desc[i1], &stat\_buf[i1]) != 0) {

if (file\_desc[i1] < 0 && comparison\_type == type\_status)

exit (2);

else

error (2, errno, "%s", file[i1]);

}

#if HAVE\_SETMODE

setmode (file\_desc[i1], O\_BINARY);

#endif

}

if (0 < same\_file (&stat\_buf[0], &stat\_buf[1])

&& file\_position (0) == file\_position (1))

exit (0);

if (comparison\_type != type\_status) {

struct stat outstat, nullstat;

if (fstat (STDOUT\_FILENO, &outstat) == 0

&& stat (NULL\_DEVICE, &nullstat) == 0

&& 0 < same\_file (&outstat, &nullstat))

comparison\_type = type\_status;

}

if (comparison\_type == type\_status

&& S\_ISREG (stat\_buf[0].st\_mode)

&& S\_ISREG (stat\_buf[1].st\_mode)) {

off\_t s0 = stat\_buf[0].st\_size - file\_position (0);

off\_t s1 = stat\_buf[1].st\_size - file\_position (1);

if (max (0, s0) != max (0, s1))

exit (1);

}

buf\_size = buffer\_lcm (STAT\_BLOCKSIZE (stat\_buf[0]),

STAT\_BLOCKSIZE (stat\_buf[1]));

for (i = 0; i < 2; i++)

buffer[i] = xmalloc (buf\_size + sizeof (word));

exit\_status = cmp ();

for (i = 0; i < 2; i++)

if (close (file\_desc[i]) != 0)

error (2, errno, "%s", file[i]);

if (exit\_status != 0 && comparison\_type != type\_status)

check\_stdout ();

exit (exit\_status);

return exit\_status;

}

/\* 比较函数主体 \*/

static int cmp () {

long line\_number = 1; /\* 记录第一次出现差异字节的行数 \*/

long char\_number = ignore\_initial + 1;

/\* Offset (1...) in files of 1st difference. \*/

size\_t read0, read1; /\* 每个文件的字节数 \*/

size\_t first\_diff; /\* 第一次出现差异的偏移地址 \*/

size\_t smaller;

char \*buf0 = buffer[0];

char \*buf1 = buffer[1];

int ret = 0;

int i;

//检查是否有需要跳过的字节

if (ignore\_initial)

for (i = 0; i < 2; i++)

if (file\_position (i) == -1) {

/\* lseek failed; read and discard the ignored initial prefix. \*/

off\_t ig = ignore\_initial;

do {

size\_t r = read (file\_desc[i], buf0, (size\_t) min (ig, buf\_size));

if (!r)

break;

if (r == -1)

error (2, errno, "%s", file[i]);

ig -= r;

} while (ig);

}

do {

read0 = block\_read (file\_desc[0], buf0, buf\_size);

if (read0 == -1)

error (2, errno, "%s", file[0]);

read1 = block\_read (file\_desc[1], buf1, buf\_size);

if (read1 == -1)

error (2, errno, "%s", file[1]);

buf0[read0] = ~buf1[read0];

buf1[read1] = ~buf0[read1];

first\_diff = (comparison\_type == type\_first\_diff

? block\_compare\_and\_count (buf0, buf1, &line\_number)

: block\_compare (buf0, buf1));

char\_number += first\_diff;

smaller = min (read0, read1);

if (first\_diff < smaller) {

switch (comparison\_type) {

case type\_first\_diff:

/\* See Posix.2 section 4.10.6.1 for this format. \*/

printf ("%s %s differ: char %lu, line %lu",

file[0], file[1], char\_number, line\_number);

if (opt\_print\_chars) {

unsigned char c0 = buf0[first\_diff];

unsigned char c1 = buf1[first\_diff];

printf (" is %3o ", c0);

printc (0, c0);

printf (" %3o ", c1);

printc (0, c1);

}

putchar ('\n');

case type\_status:

return 1;

case type\_all\_diffs:

do {

unsigned char c0 = buf0[first\_diff];

unsigned char c1 = buf1[first\_diff];

if (c0 != c1) {

if (opt\_print\_chars) {

printf ("%6lu %3o ", char\_number, c0);

printc (4, c0);

printf (" %3o ", c1);

printc (0, c1);

putchar ('\n');

} else

/\* See Posix.2 section 4.10.6.1 for this format. \*/

printf ("%6lu %3o %3o\n", char\_number, c0, c1);

}

char\_number++;

first\_diff++;

} while (first\_diff < smaller);

ret = 1;

break;

}

}

if (read0 != read1) {

if (comparison\_type != type\_status)

/\* See Posix.2 section 4.10.6.2 for this format. \*/

fprintf (stderr, "cmp: EOF on %s\n", file[read1 < read0]);

return 1;

}

} while (read0 == buf\_size);

return ret;

}

**六、程序编译、部署步骤**

在linux系统环境下使用gcc作为编译器编译源文件cmp.c，即可直接生成可执行文件

**七、运行示例**

在命令行输入命令cmp，回显如下内容

Try 'cmp --help' for more information.

创建两个文件用来测试功能，文件内容分别为abcdef以及abcdfe。输入命令cmp file1 file2得到一下回显：

file1 file2 differ: byte 4, line 1

如果两个文件相同则没有回显

接下来使用测试usage函数，输入命令cmp –help回显如下：

Mandatory arguments to long options are mandatory for short options too.

-b, --print-bytes print differing bytes

-i, --ignore-initial=SKIP skip first SKIP bytes of both inputs

-i, --ignore-initial=SKIP1:SKIP2 skip first SKIP1 bytes of FILE1 and

first SKIP2 bytes of FILE2

-l, --verbose output byte numbers and differing byte values

-n, --bytes=LIMIT compare at most LIMIT bytes

-s, --quiet, --silent suppress all normal output

--help display this help and exit

-v, --version output version information and exit

SKIP values may be followed by the following multiplicative suffixes:

kB 1000, K 1024, MB 1,000,000, M 1,048,576,

GB 1,000,000,000, G 1,073,741,824, and so on for T, P, E, Z, Y.

If a FILE is '-' or missing, read standard input.

Exit status is 0 if inputs are the same, 1 if different, 2 if trouble.

Report bugs to: bug-diffutils@gnu.org

GNU diffutils home page: <http://www.gnu.org/software/diffutils/>

General help using GNU software: <http://www.gnu.org/gethelp/>

输入命令cmp -l file1 file2再次对比文件内容回显如下：

4 144 145

5 145 146

6 146 145

**八、改进建议**

Cmp命令的输出对比内容差异，能够设置的输出格式类型较少，可以增加其类型内容，提升这个命令的使用体验。