Refo Yudhanto

CS360 Lab 6

#include <stdio.h>

#include <fcntl.h>

#include <ext2fs/ext2\_fs.h>

#include <unistd.h>

#include <string.h>

#include <stdint.h>

#define SUPERBLOCK\_MAGIC\_NUMBER 0xEF53

#define BLKSIZE 1024

#define SUPERBLK 1

#define GDBLK 2

#define FILE\_NAME\_MAX\_SIZE 256

typedef struct ext2\_super\_block SUPER;

typedef struct ext2\_group\_desc GD;

typedef struct ext2\_inode INODE;

typedef struct ext2\_dir\_entry\_2 DIR;

GD \*gp;

SUPER \*sp;

INODE \*ip;

DIR \*dp;

void get\_block(int fd, long blk, char buf[])

{

lseek(fd, blk\*BLKSIZE, 0);

read(fd, buf, BLKSIZE);

}

int super(char buf[], int fd)

{

// read SUPER block

get\_block(fd, 1, buf);

sp = (SUPER \*)buf;

printf("s\_inodes\_count = %d\n", sp->s\_inodes\_count);

printf("s\_blocks\_count = %d\n", sp->s\_blocks\_count);

printf("s\_free\_inodes\_count = %d\n", sp->s\_free\_inodes\_count);

printf("s\_free\_blocks\_count = %d\n", sp->s\_free\_blocks\_count);

printf("s\_first\_data\_blcok = %d\n", sp->s\_first\_data\_block);

printf("s\_log\_block\_size = %d\n", sp->s\_log\_block\_size);

printf("s\_blocks\_per\_group = %d\n", sp->s\_blocks\_per\_group);

printf("s\_inodes\_per\_group = %d\n", sp->s\_inodes\_per\_group);

printf("s\_mnt\_count = %d\n", sp->s\_mnt\_count);

printf("s\_max\_mnt\_count = %d\n", sp->s\_max\_mnt\_count);

printf("s\_magic = %x\n", sp->s\_magic);

}

int gd(int fd)

{

char buf[BLKSIZE];

lseek(fd, (long)GDBLK \* BLKSIZE, 0);

read(fd, buf, BLKSIZE);

GD \*gp = (GD\*)buf;

printf("bg\_block\_bitmap: %u\n", gp->bg\_block\_bitmap);

printf("bg\_inode\_bitmap: %u\n", gp->bg\_inode\_bitmap);

printf("bg\_inode\_table: %u\n", gp->bg\_inode\_table);

printf("bg\_free\_blocks\_count: %u\n", gp->bg\_free\_blocks\_count);

printf("bg\_free\_inodes\_count: %u\n", gp->bg\_free\_inodes\_count);

printf("bg\_used\_dirs\_count: %u\n", gp->bg\_used\_dirs\_count);

}

void print\_direct\_blocks(char dbuf[], const INODE \*ip, int numblk)

{

int line = 0;

for(int i = 0; i < 12; i++)

{

if(ip->i\_block[i] == 0)

return;

printf("%d ", ip->i\_block[i]);

if(++line == 10)

{

printf("\n");

line = 0;

}

numblk--;

}

numblk++;

printf("\nRemaining blocks: %d\n", numblk);

printf("\n");

}

void print\_indirect\_blocks(char dbuf[], int numblk)

{

int line = 0;

for(char\* cur = dbuf; cur < dbuf + BLKSIZE; cur += 4)

{

uint32\_t val = \*((uint32\_t\*)cur);

if(val == 0)

{

printf("\n");

printf("\nRemaining blocks: 0\n");

return;

}

printf("%d ", val);

if(++line == 10)

{

printf("\n");

line = 0;

}

numblk--;

}

numblk-=11;

printf("\nRemaining blocks: %d\n", numblk);

printf("\n");

}

void print\_double\_indirect\_blocks(int fd, char dbuf[], int numblk)

{

for(char\* cur = dbuf; cur < dbuf + BLKSIZE; cur += 4)

{

uint32\_t val = \*((uint32\_t\*)cur);

if(val == 0)

{

printf("\n");

return;

}

char dbuf2[BLKSIZE];

get\_block(fd, val, dbuf2);

print\_indirect\_blocks(dbuf2, numblk);

numblk--;

}

printf("\n");

}

void print\_blocks(int fd, const INODE \*ip)

{

for(int i = 0; i < 14; i++)

{

if(ip->i\_block[i] == 0)

break;

printf("i\_block[%d] = %d\n", i, ip->i\_block[i]);

}

int numblk = ip->i\_size / BLKSIZE;

char dbuf[BLKSIZE];

printf("================ Direct Blocks ===================\n");

print\_direct\_blocks(dbuf, ip, numblk);

if(ip->i\_block[12] != 0)

{

get\_block(fd, ip->i\_block[12], dbuf);

printf("=============== Indirect blocks ===============\n");

print\_indirect\_blocks(dbuf, numblk);

}

if(ip->i\_block[13] != 0)

{

get\_block(fd, ip->i\_block[13], dbuf);

printf("=========== Double Indirect blocks ============\n");

print\_double\_indirect\_blocks(fd, dbuf, numblk);

}

}

int search(int dev, const INODE \*ip, const char \* name)

{

printf("i\_number\trec\_len\t\tname\_len\tname\n");

char dbuf[BLKSIZE], temp[256];

DIR\* dp;

char\* cp;

for(int i = 0; i < 12; i++)

{

if(ip->i\_block[i] == 0)

break;

get\_block(dev, ip->i\_block[i], dbuf);

cp = dbuf;

dp = (DIR\*)dbuf;

while(cp < dbuf + BLKSIZE)

{

strncpy(temp, dp->name, dp->name\_len);

temp[dp->name\_len] = '\0';

printf("%4d\t\t%4d\t\t%4d\t\t%s\n",dp->inode, dp->rec\_len, dp->name\_len, temp);

if(strcmp(name, temp) == 0)

return dp->inode;

cp += dp->rec\_len;

dp = (DIR\*)cp;

}

}

return 0;

}

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

{

//check arg

if(argc != 3)

{

printf("Usage: ./a.out diskimage pathname\n");

return 0;

}

//get device

int dev = open(argv[1], O\_RDONLY);

if(dev < 0)

{

printf("Failed to open device %s\n", argv[1]);

return 1;

}

//get super

printf("\*\*\*\*\*\*\*\*\*\*\*\* super block info: \*\*\*\*\*\*\*\*\*\*\*\*\*\n");

char buf[BLKSIZE];

get\_block(dev, SUPERBLK, buf);

if(((SUPER\*)buf)->s\_magic != SUPERBLOCK\_MAGIC\_NUMBER)

{

printf("ERROR: Incorrect filesystem type.\n");

return 1;

}

super(buf,dev);

//get gd0

printf("\*\*\*\*\*\*\*\*\*\*\*\* group 0 info \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

gd(dev);

//get inode

printf("\*\*\*\*\*\*\*\*\*\*\* root inode info \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

get\_block(dev, GDBLK, buf);

int iblk = ((GD\*)buf)->bg\_inode\_table;

get\_block(dev, iblk, buf);

ip = (INODE \*)buf + 1; // ip points at 2nd INODE

printf("mode=0x%4x\n", ip->i\_mode);

printf("size=%d\n", ip->i\_size);

printf("link=%d\n", ip->i\_links\_count);

printf("i\_block[0]=%d\n", ip->i\_block[0]);

//print all root dir

get\_block(dev, 2, buf);

GD\* gp = (GD\*)buf;

get\_block(dev, gp->bg\_inode\_table, buf);

ip = (INODE\*) buf + 1;

unsigned int block0 = ip->i\_block[0];

get\_block(dev, block0, buf);

char\* cp = buf;

DIR\* dp = (DIR\*)buf;

char temp[256];

printf("\*\*\*\*\*\*\*\*\* root dir entries \*\*\*\*\*\*\*\*\*\*\*\n");

printf("i\_number\trec\_len\t\tname\_len\tname\n");

while(cp < buf + BLKSIZE)

{

strncpy(temp, dp->name, dp->name\_len);

temp[dp->name\_len] = '\0';

printf("%4d\t\t%4d\t\t%4d\t\t%s\n",

dp->inode, dp->rec\_len, dp->name\_len, temp);

cp += dp->rec\_len;

dp = (DIR\*)cp;

}

//reset root dir

printf("Press Enter to Continue\n");

getchar();

get\_block(dev, GDBLK, buf);

iblk = ((GD\*)buf)->bg\_inode\_table;

get\_block(dev, iblk, buf);

INODE\* ip = (INODE\*)buf + 1;

printf("mode=%x ", ip->i\_mode);

printf("uid=%d gid=%d\n", ip->i\_uid, ip->i\_gid);

//search for inode

char current\_dir[FILE\_NAME\_MAX\_SIZE] = "/";

char \* path = argv[2];

if(path[0] == '/')

path++;

char \* tok = strtok(path, "/");

int num;

do

{

printf("===========================================\n");

printf("Searching %s in %s \n", tok, current\_dir);

num = search(dev, ip, tok);

if(num == 0)

{

printf("NOT FOUND\n");

return 0;

}

strcpy(current\_dir, tok);

int blk = (num - 1) / 8 + iblk;

int offset = (num - 1) % 8;

get\_block(dev, blk, buf);

ip = (INODE\*)buf + offset;

printf("Found %s: ino = %d\n", current\_dir, num);

}while(tok = strtok(NULL, "/"));

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DISK BLOCKS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

print\_blocks(dev, ip);

return 0;

}

Print out Z/Hugefile:

\*\*\*\*\*\*\*\*\*\*\*\* super block info: \*\*\*\*\*\*\*\*\*\*\*\*\*

s\_inodes\_count = 184

s\_blocks\_count = 1440

s\_free\_inodes\_count = 166

s\_free\_blocks\_count = 939

s\_first\_data\_blcok = 1

s\_log\_block\_size = 0

s\_blocks\_per\_group = 8192

s\_inodes\_per\_group = 184

s\_mnt\_count = 2

s\_max\_mnt\_count = 28

s\_magic = ef53

\*\*\*\*\*\*\*\*\*\*\*\* group 0 info \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bg\_block\_bitmap: 8

bg\_inode\_bitmap: 9

bg\_inode\_table: 10

bg\_free\_blocks\_count: 939

bg\_free\_inodes\_count: 166

bg\_used\_dirs\_count: 5

\*\*\*\*\*\*\*\*\*\*\* root inode info \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

mode=0x41ed

size=1024

link=6

i\_block[0]=33

\*\*\*\*\*\*\*\*\* root dir entries \*\*\*\*\*\*\*\*\*\*\*

i\_number rec\_len name\_len name

2 12 1 .

2 12 2 ..

11 20 10 lost+found

12 12 1 X

14 12 1 Y

16 12 1 Z

18 944 4 tiny

Press Enter to Continue

mode=41ed uid=0 gid=0

===========================================

Searching Z in /

i\_number rec\_len name\_len name

2 12 1 .

2 12 2 ..

11 20 10 lost+found

12 12 1 X

14 12 1 Y

16 12 1 Z

Found Z: ino = 16

===========================================

Searching hugefile in Z

i\_number rec\_len name\_len name

16 12 1 .

2 12 2 ..

17 1000 8 hugefile

Found hugefile: ino = 17

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DISK BLOCKS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i\_block[0] = 65

i\_block[1] = 66

i\_block[2] = 67

i\_block[3] = 68

i\_block[4] = 69

i\_block[5] = 70

i\_block[6] = 71

i\_block[7] = 72

i\_block[8] = 73

i\_block[9] = 74

i\_block[10] = 75

i\_block[11] = 76

i\_block[12] = 77

i\_block[13] = 334

================ Direct Blocks ===================

65 66 67 68 69 70 71 72 73 74

75 76

Remaining blocks: 420

=============== Indirect blocks ===============

78 79 80 81 82 83 84 85 86 87

88 89 90 91 92 93 94 95 96 97

98 99 100 101 102 103 104 105 106 107

108 109 110 111 112 113 114 115 116 117

118 119 120 121 122 123 124 125 126 127

128 129 130 131 132 133 134 135 136 137

138 139 140 141 142 143 144 145 146 147

148 149 150 151 152 153 154 155 156 157

158 159 160 161 162 163 164 165 166 167

168 169 170 171 172 173 174 175 176 177

178 179 180 181 182 183 184 185 186 187

188 189 190 191 192 193 194 195 196 197

198 199 200 201 202 203 204 205 206 207

208 209 210 211 212 213 214 215 216 217

218 219 220 221 222 223 224 225 226 227

228 229 230 231 232 233 234 235 236 237

238 239 240 241 242 243 244 245 246 247

248 249 250 251 252 253 254 255 256 257

258 259 260 261 262 263 264 265 266 267

268 269 270 271 272 273 274 275 276 277

278 279 280 281 282 283 284 285 286 287

288 289 290 291 292 293 294 295 296 297

298 299 300 301 302 303 304 305 306 307

308 309 310 311 312 313 314 315 316 317

318 319 320 321 322 323 324 325 326 327

328 329 330 331 332 333

Remaining blocks: 164

=========== Double Indirect blocks ============

336 337 338 339 340 341 342 343 344 345

346 347 348 349 350 351 352 353 354 355

356 357 358 359 360 361 362 363 364 365

366 367 368 369 370 371 372 373 374 375

376 377 378 379 380 381 382 383 384 385

386 387 388 389 390 391 392 393 394 395

396 397 398 399 400 401 402 403 404 405

406 407 408 409 410 411 412 413 414 415

416 417 418 419 420 421 422 423 424 425

426 427 428 429 430 431 432 433 434 435

436 437 438 439 440 441 442 443 444 445

446 447 448 449 450 451 452 453 454 455

456 457 458 459 460 461 462 463 464 465

466 467 468 469 470 471 472 473 474 475

476 477 478 479 480 481 482 483 484 485

486 487 488 489 490 491 492 493 494 495

496 497 498 499

Remaining blocks: 0