John Dugger / xv6-pa4

Compare

Merge

master change destination

PA4 change source

Diff Commits Merged pull requests

Files changed (17)

+1	-0	M	.gitignore
+2	-1	M	Makefile
+21	-0	A	Vagrantfile
+1	-0	M	defs.h
+4	-0	M	fcntl.h
+0	-1	M	file.c
+88	-5	M	fs.c
+5	-0	M	fs.h
+154	-0	A	fsTest.c
+5	-1	M	ls.c
+9	-0	M	stat.h
+2	-0	M	syscall.c
+1	-0	M	syscall.h
+29	-5	M	sysfile.c
+1	-0	M	user.h
+1	-0	M	usys.S
+8	-0	A	vagrantprov.sh

.gitignore MODIFIED — Side-by-side diff | View file

```
14  14  kernelmemfs
15  15  mkfs
16  16  .gdbinit
    17  +.vagrant
```

Makefile MODIFIED — Side-by-side diff | View file

```
173 173         _stressfs\
174 174         _usertests\
175 175         _wc\
    176 +       _fsTest\
176 177         _zombie\
177 178
178 179 fs.img: mkfs README $(UPROGS)
```

```
243 244 EXTRA=\
244 245         mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
245 246         ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c\
246     -       printf.c umalloc.c\
    247 +       printf.c umalloc.c fsTest.c\
247 248         README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\
248 249         .gdbinit.tmpl gdbutil\
249 250
```

Vagrantfile **ADDED** — Side-by-side diff | View file

```
+# -*- mode: ruby -*-
+# vi: set ft=ruby :
+
+# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
+VAGRANTFILE_API_VERSION = "2"
+
+Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
+  config.vm.box = "ubuntu/trusty64"
+
+  # Share an additional folder to the guest VM. The first argument is
+  # the path on the host to the actual folder. The second argument is
+  # the path on the guest to mount the folder. And the optional third
+  # argument is a set of non-required options.
+  # config.vm.synced_folder "../data", "/vagrant_data"
+
+  config.vm.provider "virtualbox" do |vb|
+    vb.customize ["modifyvm", :id, "--memory", "1024"]
+  end
+
+  config.vm.provision :shell, path: "vagrantprov.sh"
+end
```

defs.h **MODIFIED** — Side-by-side diff | View file

```
...
int             fileread(struct file*, char*, int n);
int             filestat(struct file*, struct stat*);
int             filewrite(struct file*, char*, int n);
+int            lseek(int fd, int offset);

// fs.c
void            readsb(int dev, struct superblock *sb);
...
```

fcntl.h **MODIFIED** — Side-by-side diff | View file

```
...
#define O_WRONLY  0x001
#define O_RDWR    0x002
#define O_CREATE  0x200
+// add an O_EXTENT flag to the open() system call
+// that will create an extent based file
+// the number 0x201 is arbitrary
+#define O_EXTENT  0x201
```

file.c **MODIFIED** — Side-by-side diff | View file

```
...
  }
  panic("filewrite");
}
-
```

fs.c **MODIFIED** — Side-by-side diff | View file

```
...

// Return the disk block address of the nth block in inode ip.
// If there is no such block, bmap allocates one.
+
static uint
bmap(struct inode *ip, uint bn)
```

```
{
  uint addr, *a;
  struct buf *bp;
-
+  // PA4 changes: Needs to be modified to allocate memory in an extent format
+if (ip->type == T_EXTENT) {
+  uint blo = 0;
+  int i;
+  for (i = 0; i < NDIRECT + 1; i++) {
+    // (ptr,length)
+    int blockPtr = (ip->addrs[i] >> 8); //ptr for first block
+    int extentLen = (ip->addrs[i] & 0xff); //length of extent
+      if (extentLen == 0) { // Block not currently allocated
+        if (i == NDIRECT) {
+          int addr = balloc(ip->dev);
+          blockPtr = (ip->addrs[NDIRECT] >> 8);
+          extentLen = (ip->addrs[NDIRECT] & 0xff);
+          // Block adjancent to last pair i.e its the neighbouring block, we need to increase the last extent
+          if (blockPtr + extentLen == addr) {
+            ip->addrs[NDIRECT] = (blockPtr << 8) | (extentLen + 1);
+          } else {
+            panic("bmap: file exceeds maximum extents");
+          }
+        }
+        // if not adjucent to the last pair, need to allocate a new pair
+         else {
+          int addr = balloc(ip->dev);
+          if (bn == 0 && extentLen == 0) {
+            // allocating first pair and returing its address
+            ip->addrs[0] = (addr << 8) | 1;
+            return addr;
+          } else {
+            // adding another pair if this is not the first pair
+            // resetting values of first block ptr and extent length/size
+            blockPtr = (ip->addrs[bn - 1] >> 8);
+            extentLen = (ip->addrs[bn - 1] & 0xff);
+            // Now checking again if the block is adjucent to last block and coalesce it into pair
+            // if not adjucent , creating new pair in else->
+            if (blockPtr + extentLen == addr) {
+              ip->addrs[i - 1] = (blockPtr << 8) | (extentLen + 1);
+              return addr;
+            } else {
+              ip->addrs[i] = (addr << 8) | 1;
+              return addr;
+            }
+          }
+        }
+      }
+      // Returning block from this pair, if not found check next pair
+      if (bn < blo + extentLen) {
+        uint offset = bn - blo;
+        return blockPtr + offset;
+      } else {
+        blo = blockPtr + extentLen;
+      }
+    }
+  }
+  //For Direct block
  if(bn < NDIRECT){
      if((addr = ip->addrs[bn]) == 0)
        ip->addrs[bn] = addr = balloc(ip->dev);
      return addr;
    }
    bn -= NDIRECT;
-
+// for indirect block
    if(bn < NINDIRECT){
      // Load indirect block, allocating if necessary.
      if((addr = ip->addrs[NDIRECT]) == 0)
```

...

```
      brelse(bp);
      return addr;
    }
```

```
399       -
400  453      panic("bmap: out of range");
401  454  }
402  455
     ...
411  464    int i, j;
412  465    struct buf *bp;
413  466    uint *a;
414       -
     467  +  //getting length of extent
     468  +  if (ip->type == T_EXTENT) {
     469  +    int i;
     470  +    for (i = 0; i < NDIRECT + 1; i++) {
     471  +      // (ptr, length)pair
     472  +      int blockPtr = (ip->addrs[i] >> 8);
     473  +      int extentLen = (ip->addrs[i] & 0xff);
     474  +      if (extentLen != 0) {
     475  +        int j;
     476  +        for(j = 0; j < extentLen; j++) { bfree(ip->dev, blockPtr+j); }
     477  +      }
     478  +    }
     479  +  }
     480  +  else {
415  481      for(i = 0; i < NDIRECT; i++){
416  482        if(ip->addrs[i]){
417  483          bfree(ip->dev, ip->addrs[i]);
     ...
430  496        bfree(ip->dev, ip->addrs[NDIRECT]);
431  497        ip->addrs[NDIRECT] = 0;
432  498      }
433       -
     499  +  }
434  500    ip->size = 0;
435  501    iupdate(ip);
436  502  }
     ...
440  506  void
441  507  stati(struct inode *ip, struct stat *st)
442  508  {
     509  +  // fstat() uses stati to dump information. Need to also
     510  +  // modify struct stat in stat.h to hold the information
     511  +  // modify fstat() system call such that it will dump
     512  +  // information about each extent of an extent based
     513  +  // file in addition to file size
443  514    st->dev = ip->dev;
444  515    st->ino = ip->inum;
445  516    st->type = ip->type;
446  517    st->nlink = ip->nlink;
447  518    st->size = ip->size;
     519  +  // TA directed returning the address even though default
     520  +  // fstat() doesn't return the address. Added
     521  +  int i;
     522  +  for (i = 0; i < NDIRECT + 1; i++) {
     523  +    st->bloaddrs[i] = ip->addrs[i];
     524  +  }
     525  +
     526  +
448  527  }
449  528
450  529  //PAGEBREAK!
     ...
464  543
465  544    if(off > ip->size || off + n < off)
466  545      return -1;
     546  +  // if offset + > maxfilesize*block and type is not equal to T_EXTENT:
     547  +  //limits the size of each extent to 2^8 blocks and the disk addresses to 2^24.
     548  +  //if(off + n > MAXFILE*BSIZE && ip -> type != T_EXTENT)
     549  +   // n = MAXFILE*BSIZE - off;
467  550    if(off + n > ip->size)
468  551      n = ip->size - off;
469  552
```

fs.h MODIFIED

Side-by-side diff | View file

```
#define NDIRECT 12
#define NINDIRECT (BSIZE / sizeof(uint))
#define MAXFILE (NDIRECT + NINDIRECT)
+#define MAXEXTENT (6 * 256)

// On-disk inode structure
struct dinode {
```

```
  short minor;          // Minor device number (T_DEV only)
  short nlink;          // Number of links to inode in file system
  uint size;            // Size of file (bytes)
+ // keep the inode structure exactly as it
+ // is except for those data block points.
+ // Use 3 of the 4 bytes for pointer and
+ // the remaining byte for length.
  uint addrs[NDIRECT+1];   // Data block addresses
};
```

fsTest.c ADDED

Side-by-side diff | View file

```
+#include "types.h"
+#include "stat.h"
+#include "user.h"
+#include "fs.h"
+#include "fcntl.h"
+#include "syscall.h"
+#include "traps.h"
+
+int stdout = 1;
+
+int
+main (int argc, char* argv[]) {
+  int fd = open("mytestFile.txt", O_CREATE | O_RDWR | O_EXTENT);
+  int fd2 = open("mytestFile2.txt", O_CREATE | O_RDWR | O_EXTENT);
+  struct stat st, st2; //for fstat
+  if (fd <= 1) {
+    printf(stdout, "Issue creating file1\n");
+  }
+
+  if (fd2 <= 1) {
+    printf(stdout, "Issue creating file2\n");
+  }
+  printf(stdout, "**************** Starting test for file1 ******************\n");
+  int i;
+  for (i = 0; i < 65; i++) {
+    if(write(fd, "abcdef\n", 8) != 8) {
+      printf(stdout, "Error: Writing abcdef to new file1 line  %d failed\n", i);
+      exit();
+    }
+    else{printf(stdout, "Writing abcdef to line%d of file1\n", i);}
+  }
+
+  fstat(fd, &st);
+  printf(stdout, "File size i= %d\n", st.size);
+  printf(stdout, "File type = %d\n", st.type);
+  printf(stdout, "File system device = %d\n", st.dev);
+  printf(stdout, "inode num = %d\n", st.ino);
+  printf(stdout, "Number of links = %d\n", st.nlink);
+  printf(stdout, "Block Addresses:\n");
+  if (st.type == T_EXTENT) {
+    for (i = 0; i < NDIRECT + 1; i++) {
+     //first block pointer
+         int bloPtr = (st.bloaddrs[i] >> 8);
```

```
+         int len = (st.bloaddrs[i] & 0xff); // length of extent
+       printf(stdout, "  Pointer %d's address = %d with size = %d\n", i, bloPtr, len);
+     }
+   } else {
+     for (i = 0; i < NDIRECT + 1; i++) {
+       printf(stdout, "  Pointer %d's address = %d\n", i, st.bloaddrs[i]);
+     }
+   }
+   //printf(stdout, "Disk address of block = %d\n", st.bloaddrs);
+   printf(stdout, "Number of blocks = %d\n", st.length);
+   if (st.size != (512 + 8) ) {
+     printf(stdout, "File size is different from expected, size = %d\n", st.size);
+     printf(stdout, "File type = %d\n", st.type);
+   }
+
+   printf(stdout, "************* Starting test for file2 ***************\n");
+   int j;
+   for (j = 0; j < 65; j++) {
+     if(write(fd2, "hello\n", 7) != 7) {
+       printf(stdout, "Error: Writing hello to file2 line%d  failed\n", i);
+       exit();
+     }
+     else{printf(stdout, "Writing hello to line%d  of file2\n", j);}
+   }
+
+   fstat(fd2, &st2);
+   printf(stdout, "File size i= %d\n", st2.size);
+   printf(stdout, "File type = %d\n", st2.type);
+   printf(stdout, "File system device = %d\n", st2.dev);
+   printf(stdout, "inode num = %d\n", st2.ino);
+   printf(stdout, "Number of links = %d\n", st2.nlink);
+   printf(stdout, "Block Addresses:\n");
+   if (st2.type == T_EXTENT) {
+     for (j = 0; j < NDIRECT + 1; j++) {
+      //first block pointer
+         int bloPtr2 = (st2.bloaddrs[j] >> 8);
+         int len2 = (st2.bloaddrs[j] & 0xff); // length of extent
+       printf(stdout, " File2: Pointer %d's address = %d with size = %d\n", j, bloPtr2, len2);
+     }
+   } else {
+     for (j = 0; j < NDIRECT + 1; j++) {
+       printf(stdout, " File2: Pointer %d's address = %d\n", j, st2.bloaddrs[j]);
+     }
+   }
+   //printf(stdout, "Disk address of block = %d\n", st.bloaddrs);
+   printf(stdout, "Number of blocks = %d\n", st2.length);
+   if (st2.size != (448 + 7) ) {
+     printf(stdout, "File2 size is different from expected, size = %d\n", st2.size);
+     printf(stdout, "File2 type = %d\n", st2.type);
+   }
+   //Lseek testing begins here
+   int fd3;
+   if(fd3 = open("mytestFile3.txt", O_CREATE | O_RDWR | O_EXTENT) <= 1){
+     printf(stdout, "Error: Issue creating file1\n");
+   }
+   if(write(fd3, "abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ\n", 64) != 64){
+     printf(stdout, "Error: Writing to file3 failed\n");
+     exit();
+   }
+   printf(stdout, "Testing lseek(). Current file contents:\n");
+
+   // output contents of the file
+   int n;
+   char buf[512];
+   while((n = read(fd3, buf, sizeof(buf))) > 0) {
+     if (write(1, buf, n) != n) {
+       printf(1, "cat: write error\n");
+       exit();
+     }
+   }
+   if(n < 0)
+     printf(1, "cat: read error\n");
+
+   // the offset should be 64 before this
```

```
+  if (lseek(fd3, 26) == -1){
+    printf(stdout, "lseek error\n");
+  }
+  if(write(fd3, "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789", 36) != 36) {
+      printf(stdout, "Error: Writing to file3 failed\n");
+      exit();
+  }
+  // should see a gap of 10 zeroes before the last hello\n
+  printf(stdout, "Capitals and digits swapped. Current file contents:\n");
+
+  // output contents of the file
+  while((n = read(fd, buf, sizeof(buf))) > 0) {
+    if (write(1, buf, n) != n) {
+      printf(1, "cat: write error\n");
+      exit();
+    }
+  }
+  if(n < 0)
+    printf(1, "cat: read error\n");;
+
+  if(lseek(fd, -1) == 0){
+    printf(stdout, "Error: lseek() allowed negative offset\n");
+  } else{
+    printf(stdout, "lseek() correctly didn't allow a negative offset\n");
+  }
+  if(lseek(fd, 0x7FFFFFFF) == 0){
+    printf(stdout, "Error: lseek() allowed offset outside the end of the file\n");
+  } else{
+    printf(stdout, "lseek() correctly didn't allow offset outside the end of the file\n");
+  }
+
+  printf(stdout, "***********Testing Extent File system completed!*************\n");
+  close(fd);
+  close(fd2);
+  close(fd3);
+  exit();
+}
```

ls.c MODIFIED — Side-by-side diff | View file

```
   }

   switch(st.type){
-  case T_FILE:
+  case T_FILE: // or T_EXTENT
+    printf(1, "%s %d %d %d\n", fmtname(path), st.type, st.ino, st.size);
+    break;
+
+  case T_EXTENT: // handled exactly like T_FILE
     printf(1, "%s %d %d %d\n", fmtname(path), st.type, st.ino, st.size);
     break;
```

stat.h MODIFIED — Side-by-side diff | View file

```
 #define T_DIR  1   // Directory
 #define T_FILE 2   // File
 #define T_DEV  3   // Device
+// create a new type of file to support
+// inodes with pointers and length
+#define T_EXTENT 4 // Extent based file

 struct stat {
   short type;  // Type of file

   uint ino;    // Inode number
   short nlink; // Number of links to file
```

```
10  13    uint size;   // Size of file in bytes
    14  +  // adding an address after discussion with the TA
    15  +  // address wasn't included for the default fstat
    16  +  uint bloaddrs[13]; // Disk address of block
    17  +  // length will be 1 by default for T_FILE /T_DIR / T_DEV types
    18  +  // length will be the last byte of the address for T_EXTENT
    19  +  uint length; // Number of blocks
11  20    };
```

syscall.c MODIFIED — Side-by-side diff | View file | •••

```
103 103  extern int sys_wait(void);
104 104  extern int sys_write(void);
105 105  extern int sys_uptime(void);
    106 +extern int sys_lseek(void);
106 107
107 108  static int (*syscalls[])(void) = {
108 109  [SYS_fork]    sys_fork,
```

```
126 127  [SYS_link]    sys_link,
127 128  [SYS_mkdir]   sys_mkdir,
128 129  [SYS_close]   sys_close,
    130 +[SYS_lseek]   sys_lseek,
129 131  };
130 132
131 133  void
```

syscall.h MODIFIED — Side-by-side diff | View file | •••

```
20  20  #define SYS_link   19
21  21  #define SYS_mkdir  20
22  22  #define SYS_close  21
    23 +#define SYS_lseek  22
```

sysfile.c MODIFIED — Side-by-side diff | View file | •••

```
248 248    if((dp = nameiparent(path, name)) == 0)
249 249      return 0;
250 250    ilock(dp);
251     -
    251 +// checking if file is existing
252 252    if((ip = dirlookup(dp, name, &off)) != 0){
253 253      iunlockput(dp);
254 254      ilock(ip);
255     -    if(type == T_FILE && ip->type == T_FILE)
    255 +    if((type == T_FILE && ip->type == T_FILE) || (type == T_EXTENT && ip->type == T_EXTENT))
256 256        return ip;
257 257      iunlockput(ip);
258 258      return 0;
259 259    }
260 260
    261 +// creating file coz it doesnt exist
261 262    if((ip = ialloc(dp->dev, type)) == 0)
262 263      panic("create: ialloc");
263 264
```

```
297 298    begin_op();
298 299
299 300    if(omode & O_CREATE){
300     -    ip = create(path, T_FILE, 0, 0);
301     -    if(ip == 0){
302     -      end_op();
```

```
+    if (omode & O_EXTENT) {
+      if((ip = create(path, T_EXTENT, 0, 0)) == 0)
+        return -1;
+    }
+    else {
+      if((ip = create(path, T_FILE, 0, 0)) == 0)
         return -1;
     }
   } else {
```

...

```
  fd[1] = fd1;
  return 0;
}
+
+int
+sys_lseek(void)
+{
+  int fd;
+  int offset;
+  struct file *f;
+
+  // checks to verify that there are valid arguments
+  // stores the arguments in fd, f and offset
+  // makes sure the offset is within the file
+  if(argfd(0, &fd, &f) < 0 || argint(1, &offset) < 0 || offset < 0 || offset > f->ip->size)
+  {
+    return -1;
+  }
+  // f now stores the file referenced by fd
+  // changes the offset for the file to the offset value passed in
+  f->off = offset;
+  return f->off;
+}
```

user.h MODIFIED — Side-by-side diff | View file | •••

...

```
 char* sbrk(int);
 int sleep(int);
 int uptime(void);
+int lseek(int, int);

 // ulib.c
 int stat(char*, struct stat*);
```

...

usys.S MODIFIED — Side-by-side diff | View file | •••

...

```
 SYSCALL(sbrk)
 SYSCALL(sleep)
 SYSCALL(uptime)
+SYSCALL(lseek)
```

vagrantprov.sh ADDED — Side-by-side diff | View file | •••

```
+#!/usr/bin/env bash
+
+apt-get update
+apt-get install -y qemu-system-x86
+apt-get install -y gdb
+apt-get install -y tmux
+
+echo "set auto-load safe-path /" > /home/vagrant/.gdbinit
```