CS 350 Notes

Matthew Visser

Oct 20, 2011

1 Assignment 2

First things to do:

- Write to console.
- Get `_exit` working.

1.1 `_exit`

- Clean up process
  - thread
  - Address space
  - more stuff as assignment progresses like process numbers, FDs, `etc`.
- Start by looking at `thread_exit`.

1.2 Implement Argument Passing

We need programs main to get the arguments:

```
int main (int argc, char **argv);
```

Note: look at the hints page. The command to run a program and give it arguments in os/161 is the `p command arg1 arg2` command.

Issuing the `p` command calls the function in the kernel `cmd_progthread()` which calls the function `run_program()`.

For `cmd_progthread`, it has two arguments. The first pointer is to `argv`. Make sure `argv` has a `NULL` at the end of it. `argc` is the count of the number of arguments (or the last index in `argv`.

We can place `argv`:

- On the stack
- In its own segment
- somewhere crazy (as long as it works)

Look at `dumbvm` and see how it creates a stack, especially if you're putting arguments on the stack.

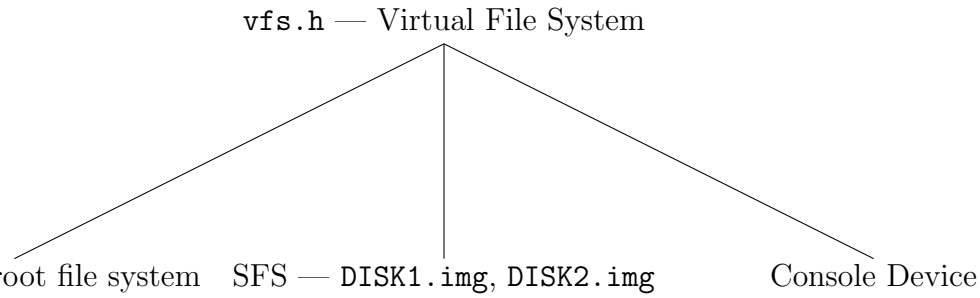
After copying `argv` and `argc`, we have to make the program aware of them by putting the arguments in `a0,a1`. To see how to do this, look at `run_program()`, and `md_usermode()`. The second function is the one that switches from kernel-mode to user-mode.

1.3 System Calls

- Files
  - `open`
  - `close`
  - `write`
- Processes
  - `fork`
  - `waitpid`
  - `_exit`
  - `execv`
- Exceptions

Do Files (without `execv` and processes first, in whatever order. Then do exceptions and `execv`.

1.3.1 File-Related System Calls



The File System Interface:

- `vfs_open( path, flags, vnode )`
  - `path` is the path to the file
  - `flags` are the read-write flags
  - `vnode` is returned, and is an abstraction of an open file.
- `vfs_close( vnode )` — closes a file

Look at the file `vnode.h` for file operations. Kernel-level file operations are done for you, but the user level file operations must be implemented. Look at slides 1 to 7 of the file system notes.

1.3.2 Process System Calls

Need to write `md_forkentry`.

2 Address Space

2.1 Loading a Program into Address Space — ELF Files

- Format for executables is called ELF. It is used by the kernel to create the address space for a process.
- And ELF file contains
  - An *ELF Header*
  - Program headers
    - Contains one header per segment.
    - The header contains the following:
      - Virtual address of segment
      - Length of the segment
      - Location of the start of the image in ELF file
      - Length of the image in ELF file.
  - *Segments* $1 \ldots n$.
  - *Section Headers*
- A *segment* is a chunk of data to be loaded into the address space.
- Kernel must fill rest of segment with 0's.
- OS/161 expects the ELF file to have two segments:
  - Program code
  - Data

2.2 ELF Sections and Segments

- The ELF file contains different sections
  - `.text` — program code
  - `.rodata` — read-only global data

  - `.data` — initialized global data
  - `.bss` — uninitialized global data
  - `.sbss` — small uninitialized global data
- Note all of these are present in every ELF file.
- `.text` and `.rodata` for the `text` segment. The rest for the `data` segment.

¡++¿