OSM
G assignment 5

Tobias Hallundbæk Petersen (xtv657)
Ola Rønning (vdl761)
Nikolaj Høyer (ctl533)

March 17th, 2014

Contents

1 System calls for the Buenos file system

We have extended the syscall cases in `proc/syscall.c` with the syscalls shown below:

```
    case SYSCALL_OPEN:
      V0 = (int) vfs_open((char *)A1) + 3;
      break;
    case SYSCALL_CLOSE:
      V0 = vfs_close(A1 - 3);
      break;
    case SYSCALL_SEEK:
      V0 = vfs_seek(A1 - 3, A2);
      break;
    case SYSCALL_CREATE:
      V0 = vfs_create((char *)A1, A2);
      break;
    case SYSCALL_DELETE:
      V0 = vfs_remove((char *)A1);
      break;
```

Moreover, we have modified the `syscall_write` and `syscall_read` functions, `proc/syscall_write` is shown below:

```
int syscall_write(uint32_t fd, char *s, int len)
{
  gcd_t *gcd;
  device_t *dev;
  if (fd == FILEHANDLE_STDOUT || fd == FILEHANDLE_STDERR) {
    dev = device_get(YAMS_TYPECODE_TTY, 0);
    gcd = (gcd_t *)dev->generic_device;
    return gcd->write(gcd, s, len);
  } else {
    return vfs_write(fd - 3, s, len);
  }
}
```

The functions have been altered to check the `fd` parameter and execute the existing `gcd` write if we want to write to `stdout` or `stderr`. Otherwise we interpret fd as ... and execute a `vfs_write`. `fd` represents and we -3 because...

The same applies to `syscall_read`. Here we only need to check if `fd` is set to `stdin`, since we can't write to `stderr`.

2 A simple shell and directory listing support

Firstly, we have extended the supplied `BUENOS` shell `osh.c` with the function `cmd_exit`:

```
int cmd_exit() {
  syscall_exit();
  return 0;
}
```

The function is very simple - it utilises the exit syscall defined in `proc/syscall.c` to finish the current process, and returns `0` on success.

Next, we define the `cmd_rm` function to call the `syscall_delete` function defined in task 1 with the input pathname.

```
int cmd_help() {
  help();
  return 0;
```