The basics of programming in C

```
struct ext2_fs
{
    int fd;
    uint32_t block_size;
    …

struct ext2_blkiter
{
    int fd;
    uint32_t block_size;
    …
```

```
struct ext2_fs
{
    int fd;
    uint32_t block_size;
    …

struct ext2_blkiter
{
    int fd;
    uint32_t block_size;
    …
```

```
struct ext2_blkiter
{
    struct ext2_fs *fs;
    …
```

```
int ext2_fs_init(struct ext2_fs **fs, int fd)
{
    *fs = fs_xmalloc(sizeof(struct ext2_fs));
    (*fs)->fd = fd;

    struct ext2_super_block sb;
    if (pread(fd, &sb, sizeof(sb), 1024) < 0) {
        return -errno;
    }
    …
```

```
int ext2_fs_init(struct ext2_fs **fs, int fd)
{
    *fs = fs_xmalloc(sizeof(struct ext2_fs));
    (*fs)->fd = fd;

    struct ext2_super_block sb;
    if (pread(fd, &sb, sizeof(…), 1024) < 0) {
        return -errno;
    }
    …
```

```
int ext2_fs_init(struct ext2_fs **fs, int fd)
{
    int r;

    *fs = fs_xmalloc(sizeof(*fs));
    (*fs)->fd = fd;

    struct ext2_super_block sb;
    if (pread(fd, &sb, sizeof(sb), 1024) < 0) {
        r = -errno;
        goto out_err;
    }
    …

    return 0;
out_err:
    fs_xfree(*fs);
    return r;
}
```

```
static int read_exact_at(int fd, void *buf, count,
offset)
{
    uint8_t *p = (uint8_t *)buf;
    size_t remaining = count;

    while (remaining > 0) {
        ssize_t r = pread(fd, p, remaining, offset);
        p += (size_t)r;
        remaining -= (size_t)r;
        offset += (off_t)r;
    }
    …
```

```
static int read_exact_at(int fd, void *buf, count,
offset)
{
    uint8_t *p = (uint8_t *)buf;
    size_t remaining = count;

    while (remaining > 0) {
        ssize_t r = pread(fd, p, remaining, offset);
        p += (size_t)r;
        remaining -= (size_t)r;
        offset += (off_t)r;
    }
    …
```

```
static int read_exact_at(int fd, void *buf,
count, offset)
{
    uint8_t *p = buf;
    …
```

```
static int read_exact_at(int fd, void *buf,
count, offset)
{
    return pread(fd, buf, count, offset);
}
```

```
if (total_blocks > 0) {
    it->blocks = malloc(total_blocks * sizeof(int));
    if (!it->blocks) {
        free(it);
        return -ENOMEM;
    }

    for (int j = 0; j < total_blocks; ++j) {
        uint32_t b;
        r = inode_get_block(fs, &inode, j, &b);
        …
```

```
int dump_file(int img, int inode_nr, int out)
{
    struct ext2_fs *fs;
    struct ext2_blkiter *it;
    int r;
    if ((r = ext2_fs_init(&fs, img)) < 0) {
        return r;
    }
    if ((r = ext2_blkiter_init(&it, fs, inode_nr)) < 0) {
        ext2_fs_free(fs);
        return r;
    }
    …
         if (pread(img, buf, fs->block_size, off) < 0) {
             ext2_blkiter_free(it);
             ext2_fs_free(fs);
             return -errno;
         }
   …
   }
   ext2_blkiter_free(it);
   ext2_fs_free(fs);
   return 0;
}
```

```
int dump_file(int img, int inode_nr, int out)
{
    struct ext2_fs *fs;
    struct ext2_blkiter *it;
    int r;
    if ((r = ext2_fs_init(&fs, img)) < 0) {
        return r;
    }
    if ((r = ext2_blkiter_init(&it, fs, inode_nr)) < 0) {
        ext2_fs_free(fs);
        return r;
    }
    …
         if (pread(img, buf, fs->block_size, off) < 0) {
             ext2_blkiter_free(it);
             ext2_fs_free(fs);
             return -errno;
         }
   …
   }
   ext2_blkiter_free(it);
   ext2_fs_free(fs);
   return 0;
}
```

```
int dump_file(int img, int inode_nr, int out)
{
    struct ext2_fs *fs = NULL;
    struct ext2_blkiter *it = NULL;
    int r;
    if ((r = ext2_fs_init(&fs, img)) < 0) {
        return r;
    }
    if ((r = ext2_blkiter_init(&it, fs, inode_nr)) < 0) {
        goto out;
    }
    …
         if (pread(img, buf, fs->block_size, off) < 0) {
             r = -errno;
             goto out;
         }
   …
   }
out:
   ext2_blkiter_free(it);
   ext2_fs_free(fs);
   return r;
}
```

```
int dump_file(int img, int inode_nr, int out)
{
    struct ext2_fs *fs;
    struct ext2_blkiter *it;
    int r;
    if ((r = ext2_fs_init(&fs, img)) < 0) {
        return r;
    }
    if ((r = ext2_blkiter_init(&it, fs, inode_nr)) < 0) {
        ext2_fs_free(fs);
        return r;
    }
    …
         if (pread(img, buf, fs->block_size, off) < 0) {
             ext2_blkiter_free(it);
             ext2_fs_free(fs);
             return -errno;
         }
   …
   }
   ext2_blkiter_free(it);
   ext2_fs_free(fs);
   return 0;
}
```

```
int dump_file(int img, int inode_nr, int out)
{
    struct ext2_fs *fs = NULL;
    struct ext2_blkiter *it = NULL;
    int r;
    if ((r = ext2_fs_init(&fs, img)) < 0) {
        return r;
    }
    if ((r = ext2_blkiter_init(&it, fs, inode_nr)) < 0) {
        goto out;
    }
    …
         if ((r = ext2_read_block(fs, buf, blknr) < 0)) {
             goto out;
         }
   …
   }
out:
   ext2_blkiter_free(it);
   ext2_fs_free(fs);
   return r;
}
```