Writing programs

Return values
Arguments
Writing Programs

Return values

• In a standard C environment, there is an Operating System
• Programs are started, execute, and end within the OS
• The return value allows for a program to return a code indicating its operation
• Most useful when writing daemons or programs that are not directly executed by the user
Writing Programs

Return values

- Returning 0 indicates successful operation
- Returning non-zero indicates error

Example

```c
int main(void)
{
    return 0;
}
```
Writing Programs

Return values

- `<stdlib.h>` defines `EXIT_SUCCESS` and `EXIT_FAILURE`

Example

```c
int main(void)
{
    return EXIT_SUCCESS;
}
```
Writing Programs
Return values

Syntax

```c
void exit(int status);
```

- Defined in `<stdlib.h>`

Example

```c
int main(void)
{
    exit(EXIT_FAILURE);
    return EXIT_SUCCESS;
}
```
Writing Programs

Program arguments

• Programs can take a variable number of arguments
  – Just like functions

• The number of arguments is known

• Only makes sense in a multi-process environment
  – Doesn't work with XC32
Writing Programs

Program arguments

Syntax

```c
int main(int argc, char *argv[]);
```

- Arguments are passed as strings
- First argument is the program name

Example

```
ls -hal ~
mkdir .ssh
ln -s ~/Dropbox/config/ .ssh .ssh
```
Writing Programs

Program arguments

```
ln -s ~/Dropbox/config/.ssh .ssh
```

4

```
argc
argv
```

Syntax

```
int main(int argc, char *argv[]);
```
int main(int argc, char *argv[])
{
    int i;
    for (i = 0; i < argc; ++i) {
        printf("\%s ", argv[i]);
    }

    return EXIT_SUCCESS;
}
CMPE-013/L

Introduction to “C” Programming

Maxwell James Dunne

Spring 2015
File I/O
File I/O

- Most data on computers are stored in files
- So accessing data reads and writes to these files
- And in a Unix environment, everything is a file
  - Serial ports
  - Network connections
  - Hard drives
  - Displays
- So everything can be controlled via file access
File I/O

Standard files

- Three special files that are automatically opened and closed
  - stdin: standard input (keyboard/serial port)
  - stdout: standard output (screen)
  - stderr: standard error (screen)
File I/O
The standard library

- `<stdio.h>` contains functions for working with files
- Its concept of a file includes:
  - Filename
  - File access mode
  - File size
  - Current position
File I/O
Using files

- Files are opened with `fopen()`
- Files are read and written to:
  - `fprintf()`, `fscanf()` — Formatting strings
  - `fputc()`, `fgetc()` — Characters
  - `fputs()`, `fgets()` — Lines
  - `fread()`, `fwrite()` — Blocks
- Files are closed with `fclose()`
File I/O

Using files

- Only a limited number of files can be opened at a time
  - Per process
  - Also per OS
- Very large on modern Oses
  - \(\geq 2048\) usually
- For the XC32: 8
The standard library uses a single struct to store the metadata of the file.

```c
typedef struct _iobuf {
    char * _ptr;
    int _cnt;
    char * _base;
    unsigned short _flag;
    short _file;
    size_t _size;
} FILE;
```
File I/O

`fopen()`

Syntax

```c
FILE *fopen(const char *name, const char *mode);
```

- **name** is a C string with the filename
- **mode** is the mode to open the file in
  - "r" opens for reading
  - "w" opens for writing
  - "a" opens for appending
  - "b" specifies binary
- Returns the file pointer
## File I/O

### File modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>r</code></td>
<td>Open a text file for reading.</td>
</tr>
<tr>
<td><code>w</code></td>
<td>Truncate to zero length or create a text file for writing.</td>
</tr>
<tr>
<td><code>a</code></td>
<td>Append; open or create a text file for writing at the end-of-file.</td>
</tr>
<tr>
<td><code>rb</code></td>
<td>Open a binary file for reading.</td>
</tr>
<tr>
<td><code>wb</code></td>
<td>Truncate to zero length or create a binary file for writing.</td>
</tr>
<tr>
<td><code>ab</code></td>
<td>Append; open or create a binary file for writing at the end-of-file.</td>
</tr>
<tr>
<td><code>r+</code></td>
<td>Open a text file for read/write.</td>
</tr>
<tr>
<td><code>w+</code></td>
<td>Truncate to zero length or create a text file for read/write.</td>
</tr>
<tr>
<td><code>a+</code></td>
<td>Append; open or create a text file for read/write. You can read data anywhere in the file, but you can write data only at the end-of-file.</td>
</tr>
<tr>
<td><code>r+b</code> or <code>rb+</code></td>
<td>Open a binary file for read/write.</td>
</tr>
<tr>
<td><code>w+b</code> or <code>wb+</code></td>
<td>Truncate to zero length or create a binary file for read/write.</td>
</tr>
<tr>
<td><code>a+b</code> or <code>ab+</code></td>
<td>Append; open or create a binary file for read/write. You can read data anywhere in the file, but you can write data only at the end-of-file.</td>
</tr>
</tbody>
</table>
File I/O

fread()

Syntax

```c
size_t fread(void *ptr, size_t size,
             size_t count, FILE *stream);
```

- `ptr` – The buffer to write into
- `size` – The size of each element to read
- `count` – The number of elements to read
- `stream` – The pointer to the file

- Returns the number of elements read
  - Less than `count` indicates error or EOF
File I/O
fread()

Syntax

```c
size_t fread(void *ptr, size_t size,
             size_t count, FILE *stream);
```

- **ptr** – The buffer to write into
- **size** – The size of each element to read
- **count** – The number of elements to read
- **stream** – The pointer to the file

- Returns the number of elements read
  - Less than count indicates error or EOF
File I/O

feof()

Syntax

```
int feof(FILE *stream);
```

- **stream** – The pointer to the file
- Returns a non-zero value if the stream is at the end of the file, 0 otherwise
File I/O

fseek()

Syntax

```c
int fseek(FILE *stream, long offset, int origin);
```

- **stream** – The pointer to the file
- **offset** – The bytes to move from the current location
- **origin** – The reference location: either SEEK_SET, SEEK_CUR, or SEEK_END
- Returns 0 if successful, otherwise returns a non-zero value
File I/O

fclose()

Syntax

```c
int fclose(FILE *stream);
```

- `stream` – The pointer to the file
- Returns 0 if successful, otherwise returns EOF
  – EOF is a macro, generally -1
int main(void)
{
    // Open the file, terminating if there was an error
    FILE *pFile = fopen("/room1.txt", "rb");
    if (pFile == NULL) {
        puts("Error opening file.");
        return EXIT_FAILURE;
    }

    // Count the characters in the file.
    int n = 0;
    while (fgetc(pFile) != EOF) {
        ++n;
    }

    // Output the results, if we succeeded
    if (feof(pFile)) {
        printf("Total bytes read: %d\n", n);
        fclose(pFile);
        return EXIT_SUCCESS;
    }

    // Otherwise output an error
    puts("Error occurred before reading end of file.");
    fclose(pFile);
    return EXIT_FAILURE;
}
File formats
File formats

Types

- Two groups:
  - Text
  - Binary

- Text are easier to process, but larger
- Binary are harder to process, but smaller
- Many formats are now zipped text files so the data is easy to parse, but the size is small
  - .docx/.xlsx for example

255 36 bytes
0xF 16 bytes

33415, 67823
File formats

Text: XML

<MetaData>
  <messageInfo name = "System Time" pgn = "126992" size = "8">
    <desc>Represents the current data and time</desc>
    <field
      name = "Days since epoch"
      type = "int"
      offset = "16"
      length = "16"
      signed = "no"
      units = "days"
      endian = "little"
    />
  </messageInfo>
  <messageInfo name = "Rudder" pgn = "127245" size = "6">
    <desc>Represents the current rudder position</desc>
    <field
      name = "Position"
      type = "int"
      offset = "32"
      length = "16"
      signed = "yes"
      units = "rad"
      scaling = "0.0001"
      endian = "little"
    />
  </messageInfo>
</MetaData>
### File formats

**Text:** CSV

<table>
<thead>
<tr>
<th>timestamp, time_usec, fix_type, lat, lon, alt, eph, env, vel, cog</th>
</tr>
</thead>
<tbody>
<tr>
<td>57.3000000000000004, 57300, 450, -59, -15857, 0.0, 0.0, 0.0, 0.0, 3, 369640780, -1220013611, 0, 150, 159, 1, 13186</td>
</tr>
<tr>
<td>57.5500000000000004, 57550, 457, -51, -15855, 0.0, 0.0, 0.0, 0.0, 3, 369640785, -1220013613, 0, 149, 159, 1, 13411</td>
</tr>
<tr>
<td>57.8000000000000004, 57800, 468, -42, -15854, 0.0, 0.0, 0.0, 0.0, 3, 369640786, -1220013615, 0, 149, 159, 1, 13458</td>
</tr>
<tr>
<td>58.0500000000000004, 58050, 474, -32, -15850, 0.0, 0.0, 0.0, 0.0, 3, 369640788, -1220013615, 0, 149, 159, 2, 13620</td>
</tr>
<tr>
<td>58.3000000000000004, 58300, 477, -17, -15847, 0.0, 0.0, 0.0, 0.0, 3, 369640788, -1220013615, 0, 149, 159, 2, 13620</td>
</tr>
<tr>
<td>58.5500000000000004, 58550, 474, -9, -15846, 0.0, 0.0, 0.0, 0.0, 3, 369640793, -1220013616, 0, 150, 159, 1, 13607</td>
</tr>
<tr>
<td>58.8000000000000004, 58800, 469, -12, -15843, 0.0, 0.0, 0.0, 0.0, 3, 369640796, -1220013616, 0, 150, 159, 1, 13616</td>
</tr>
<tr>
<td>59.0500000000000004, 59050, 468, -18, -15839, 0.0, 0.0, 0.0, 0.0, 3, 369640798, -1220013618, 0, 150, 159, 1, 13486</td>
</tr>
<tr>
<td>59.3000000000000004, 59300, 471, -14, -15841, 0.0, 0.0, 0.0, 0.0, 3, 369640798, -1220013618, 0, 150, 159, 1, 13486</td>
</tr>
<tr>
<td>59.5500000000000004, 59550, 485, -4, -15836, 0.0, 0.0, 0.0, 0.0, 3, 369640803, -1220013618, 0, 150, 159, 2, 13468</td>
</tr>
<tr>
<td>59.8000000000000004, 59800, 501, 0, -15833, 0.0, 0.0, 0.0, 0.0, 3, 369640804, -1220013618, 0, 150, 159, 1, 13441</td>
</tr>
<tr>
<td>60.0500000000000004, 60050, 502, -18, -15839, 0.0, 0.0, 0.0, 0.0, 3, 369640808, -1220013618, 0, 150, 159, 2, 13313</td>
</tr>
<tr>
<td>60.3000000000000004, 60300, 507, -28, -15839, 0.0, 0.0, 0.0, 0.0, 3, 369640808, -1220013618, 0, 150, 159, 2, 13300</td>
</tr>
<tr>
<td>60.5500000000000004, 60550, 504, -25, -15824, 0.0, 0.0, 0.0, 0.0, 3, 369640815, -1220013620, 0, 149, 159, 1, 12704</td>
</tr>
<tr>
<td>60.8000000000000004, 60800, 515, -20, -15824, 0.0, 0.0, 0.0, 0.0, 3, 369640818, -1220013620, 0, 150, 159, 2, 12492</td>
</tr>
<tr>
<td>61.0500000000000004, 61050, 524, -14, -15832, 0.0, 0.0, 0.0, 0.0, 3, 369640823, -1220013621, 0, 149, 159, 1, 12492</td>
</tr>
<tr>
<td>61.3000000000000004, 61300, 518, -7, -15844, 0.0, 0.0, 0.0, 0.0, 3, 369640823, -1220013621, 0, 149, 159, 1, 12492</td>
</tr>
<tr>
<td>61.5500000000000004, 61550, 512, 0, -15825, 0.0, 0.0, 0.0, 0.0, 3, 369640830, -1220013623, 0, 150, 159, 5, 11498</td>
</tr>
<tr>
<td>61.8000000000000004, 61800, 494, 0, -15825, 0.0, 0.0, 0.0, 0.0, 3, 369640833, -1220013623, 0, 150, 159, 2, 11094</td>
</tr>
<tr>
<td>62.0500000000000004, 62050, 485, -1, -15824, 0.0, 0.0, 0.0, 0.0, 3, 369640836, -1220013623, 0, 149, 159, 1, 11094</td>
</tr>
</tbody>
</table>
File formats

Binary: ZIP

Relative offset 1
Relative offset 2
Relative offset 3
Relative offset n

FILE ENTRY 1 <data>
FILE ENTRY 2 <data>
FILE ENTRY 3 <data>
FILE ENTRY 4 <data>
CENTRAL DIRECTORY

Local header 1
Local header 2
Local header 3
Local header n

File entry 1
File entry 2
File entry 3
File entry n

Maxwell James Dunne – Spring 2015
CMPE-013/L: “C” Programming
File formats

Binary: RPG

- Needed a format to store each room in a dungeon
- Requirements
  - Title
  - Description
  - Items in the room
  - Exits:
    - Which room
    - What direction

II) NESW
## File formats

Binary: RPG

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
</table>

Maxwell James Dunne – Spring 2015
File formats

Binary: RPG

- But it would be cool if the rooms could change depending on items the player has encountered
  - Like keys

- So we want different versions of the room for:
  - Description
  - Items
  - Exits
# File formats

**Binary: RPG**

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(repeated)
## File formats

Binary: RPG

<table>
<thead>
<tr>
<th>size</th>
<th>ASCII data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Title

Item requirements

Description

Items contained

Exits
File formats

Binary: RPG

<table>
<thead>
<tr>
<th>size</th>
<th>binary data</th>
</tr>
</thead>
</table>

- Title
- Item requirements
- Description
- Items contained
- Exits
File formats

Binary: RPG

<table>
<thead>
<tr>
<th>size</th>
<th>ASCII data</th>
</tr>
</thead>
</table>

- Title
- Item requirements
- Description
- Items contained
- Exits
File formats
Binary: RPG

<table>
<thead>
<tr>
<th>size</th>
<th>binary data</th>
</tr>
</thead>
</table>

| Title | Item requirements | Description | Items contained | Exits |

Maxwell James Dunne – Spring 2015
File formats

Binary: RPG
### File formats

**Binary: RPG**

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Version 1:</strong> Requires key, no items</td>
<td></td>
<td></td>
<td></td>
<td><strong>Version 2:</strong> No requirements, contains key</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/Room32.txt
File formats

Binary: RPG

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 The Throne Room</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Version 1: Requires key, no items

Version 2: No requirements, contains key

/Room32.txt
**File formats**

**Binary: RPG**

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Version 1:** Requires key, no items

**Version 2:** No requirements, contains key

/Room32.txt

---

*Maxwell James Dunne – Spring 2015*
File formats

Binary: RPG

169 A large metal throne forged of swords of previous kings sits prominently here. Your dad is rarely in it, however, instead ruling the kingdom from his council’s chambers.

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1:</td>
<td>Requires key, no</td>
<td>Contains key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version 2:</td>
<td>No requirements,</td>
<td>Contains key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/Room32.txt
**File formats**

Binary: RPG

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1:</td>
<td>Requires key, no items</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version 2:</td>
<td>No requirements, contains key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/Room32.txt
# File formats

**Binary: RPG**

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1: Requires key, no items</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 2: No requirements, contains key</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/Room32.txt
## File formats

**Binary: RPG**

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Version 1:** Requires key, no items

**Version 2:** No requirements, contains key

/Room32.txt
A large metal throne forged of swords of previous kings sits prominently here. Your dad is rarely in it, however, instead ruling the kingdom from his council's chambers. You feel the weight of the castle key stolen earlier in your pocket.

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1: Requires key, no items</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version 2: No requirements, contains key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

/Room32.txt
## File formats

**Binary: RPG**

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Version 1:** Requires key, no items
- **Version 2:** No requirements, contains key

/Room32.txt
File formats

Binary: RPG

<table>
<thead>
<tr>
<th>Title</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
<th>Item requirements</th>
<th>Description</th>
<th>Items contained</th>
<th>Exits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Version 1: Requires key, no items

Version 2: No requirements, contains key

/Room32.txt
\[ \text{In} \]

$\text{SHIFT, 3, 4, 1} \times 4e$

\[ H = \sum_{n \geq 1} \]

iterate string from 2 on:

\[ H^\wedge = \sum_{n \geq 1} \]

\[ t \]
Guess Data

32-t row: 8
col: Hi-status:
32 5vegg 16  rand()
    enkey 16
    en guess 16
hash  8
guess ^ enkey

H = (vin+8-t) (guess >> 8)
H ^= (vin+8-t) (guess);
\[
\text{Board_GetPBCLK()} / 8
\]

\[
\text{for } (i = 0; i < t; i++)
\]

\[
\text{--- asm ("Nop")};
\]

3
guess
row
col
Hit
$HIT, 0, 0, 1^4c$

Agent → run

Prot Decode
Field
Field Init  Visual
Field Add Boat  Valid in valid
Field AT
Field Set Location
Protocol

use given message

Invalid message
Invalid checksum

Encode with function
then Decode
5rand(1)
gen N
5rand(2)