# Hosted C and compiling C code with Linux

Adam Boult (www.bou.lt)

April 30, 2025

# Contents

# Preface

This is a live document, and is full of gaps, mistakes, typos etc.

# Part I

# GNU binutils

# Chapter 1

# GNU Binutils, including as (assembler), ld (linker), gprof (profiler), objdump

## 1.1 Introduction

### 1.1.1 Introduction

# Part II

# Compiling

# Chapter 2

# GNU Compiler Collection (gcc) and the GNU C Compiler (also gcc)

## 2.1 Introduction

### 2.1.1 Introduction

compile to object,

compile to assembly, links and runtime libraries? or is in make?

objdump; returning stderr and stdout. raise

# Chapter 3

# C headers for linux

## 3.1 Introduction

### 3.1.1 Introduction

### 3.1.2 getopt

```
#include <unistd.h>

int getopt(int argc, char *const argv[],
const char *optstring);

extern char *optarg;
extern int optind, opterr, optopt;
```

allows

```
myScript -a s
```

### 3.1.3 Longer

```
#include <getopt.h>

int getopt_long(int argc, char *const argv[],
           const char *optstring,
           const struct option *longopts, int *longindex);
int getopt_long_only(int argc, char *const argv[],
           const char *optstring,
           const struct option *longopts, int *longindex);
```

allows

```
myScript --a something
```

# Chapter 4

# make

## 4.1 Introduction

### 4.1.1 Introduction

Makefile; target:, .PHONY clean; clean: (means that doesn't look to see if clean exists because declared as phony.)

# Part III

# Hosted C: C standard library: stdlib.h

# Chapter 5

# stdlib.h: Maths (abs, labs, div_t, ldiv_t)

## 5.1 Introduction

### 5.1.1 Introduction

int abs(int x)

long int labs(long x);

div_t div(int numer, int denom); ldiv_t ldiv(long int numer, long int denom);

div_t and ldiv_t types are introduced in stdlib.h

# Chapter 6

# stdlib.h: abort() and exit()

## 6.1 Introduction

### 6.1.1 Introduction

void abort(void); void exit(int status);

# Chapter 7

# stdlib.h: Type conversions (atof, atol, strtod, strtol, strtoul)

## 7.1 Introduction

### 7.1.1 Introduction

character stuff with stdlib.h double atof int atoi long int atol long strtod long int strtol usigned long int strtoul

# Chapter 8

# stdlib.h: The heap

## 8.1 Introduction

### 8.1.1 Introduction

variable length array using the heap or the stack

```
void my_function(int n){
    float vals[n];
    return process(n, vals);
}
```

stack approach above. deleted when function ends

```
void my_function(int n){
    [something using malloc]
}
```

heap memory: allocated at run time needs to be manually removed

stack overflow + stack hits point reserved for heap?

+ memory leaks

space for stack limitated with need to reserve space to prevent overlapping with heap heap in c doesn't mean data structure. heap as in pile, or pool, just a description. just synonym for pile, area etc. Heap in c. Allocate with malloc?

### 8.1.2 malloc

malloc (memory allocation):

```
int * ptr = (int *) malloc(10 * sizeof(int));
```

### 8.1.3 calloc

calloc (contiguous allocation):

```
int * ptr = (int *) calloc(10, sizeof(int));
```

calloc initiates all to 0

### 8.1.4 free

can free

```
int * ptr = (int *) malloc(10 * sizeof(int));
free(ptr);
```

or

```
int * ptr = (int *) calloc(10, sizeof(int));
free(ptr);
```

### 8.1.5 realloc

```
int * ptr = (int *) malloc(10 * sizeof(int));
ptr = realloc(ptr, 20*sizeof(int));
```

# Chapter 9

# stdlib.h: Algorithms (qsort, bsearch, rand)

## 9.1 Introduction

### 9.1.1 Introduction

### 9.1.2 qsort()

sort in standard library (qsort)

### 9.1.3 RAND

```
void srand(unsigned int seed);
int rand(void); (returns between 0 and RAND_MAX)
```

### 9.1.4 bsearch

```
void *bsearch()
```

# Part IV

# Hosted C: C standard library

# Chapter 10

# math.h, complex.h and tgmath.h

## 10.1 Introduction

### 10.1.1 math.h

### 10.1.2 complex.h

Complex numbers.

### 10.1.3 tgmath.h

Type-generic math.

# Chapter 11

# fenv.h

## 11.1 Introduction

### 11.1.1 Introduction

# Chapter 12

# setjmp.h

## 12.1 Introduction

### 12.1.1 Introduction

# Chapter 13

# Getting and setting locale using locale.h (eg keyboard, date format)

## 13.1 Introduction

### 13.1.1 Introduction

# Chapter 14

# Time and date with time.h

## 14.1 Introduction

### 14.1.1 Introduction

Date or time handling.

# Chapter 15

# Working with strings with string.h and ctype.h

## 15.1 Introduction

### 15.1.1 string.h

```
string.h Functions on strings using
void * memchr
+ find character in string, return pointer
int memcmp
+ are 2 blocks of strings the same
void * memcpy
+ copy n  characters from sour e to dest
void * memmove
+ also copy n characters
void * memset
+ replace first n characters of string by a single character
char * strcat
+ concatenate 2 strings
char * strncat
+ concatenate 2 strings, first n characters
char * strchr
+ find character in string
int strcmp
+ compare 2 strings
int strncmp
+ compare first n characters of 2 strings
int strcoll
+ compare 2 strings according to locale
```

```
char * strcpy
+ copy a string
char * strncpy
+ copy first n characters of string
size_t strcspn
+ return span (number of characters) in string 1 until found any matching characters in stri
char * strerror
+ returns pointer to string with error
size_t strlen
+ return length of string
char * strpbrk
+ like strcspn but returns pointer
char * strrchr
+ find last instance of character in string
size_t strspn
+ like strcspn but until found character any not matching characters in string 2
char * strstr
+ find first instance of substring *needle in string *haystack
char * strtok
+ breaks string up using delimiters
size_t strxfrm
+ copy part of a string and convert using locale
```

# Chapter 16

# Specified width integers with inttypes.h (replacing freestanding C's stdint.h)

## 16.1 Introduction

### 16.1.1 Introduction

# Chapter 17

# Errors and signals with errno.h, signal.h and assert.h

## 17.1 Introduction

### 17.1.1 Introduction

# Chapter 18

# Wide characters including Unicode with uchar.h, wchar.h and wctype.h

## 18.1 Introduction

### 18.1.1 Introduction

# Chapter 19

# stdio.h

## 19.1 Introduction

### 19.1.1 Introduction

### 19.1.2 stdio

stdio allows arguments but not as flags. allows

```
myScript something
```

# Chapter 20

# Multi-threading with threads.h and stdatomic.h

## 20.1 Introduction

### 20.1.1 Introduction

# Chapter 21

# ctype.h

## 21.1 Introduction

### 21.1.1 Introduction

# Part V

# Analysing programs

# Chapter 22

# Static analysis

## 22.1 Introduction

### 22.1.1 Introduction

frama-C?, formal verification

# Chapter 23

# Dynamic analysis

## 23.1 Introduction

### 23.1.1 Introduction

unit tests (inc asserts) and integration tests

# Chapter 24

# GNU Debugger (gdb)

## 24.1 Introduction

### 24.1.1 Introduction

debugger. can step through, stop on specific functions etc