The Standard C Library

http://www.freshsources.com

Agenda

- Group I: For the Adequate Programmer
- Group II: For the Polished Programmer
- Group III: For the Complete Programmer

Group I

Group I (required knowledge for every C programmer)

<ctype.h>	Character Handling
<stdio.h>	Input/Output
<stdlib.h>	Miscellaneous Utilities
<string.h>	Text Processing

<ctype.h>

isalnum	alphanumeric (isalpha || isdigit)
isalpha	alphabetic
iscntrl	control (beware!)
isdigit	'0' through '9'
isgraph	visible when printed
islower	lower case alphabetic
isprint	isgraph || ' '
ispunct	isgraph && !isalnum
isspace	whitespace
isupper	upper case alphabetic
isxdigit	isdigit || 'a' thru 'f' || 'A' thru 'F'
tolower	convert to lower case (if applicable)
toupper	convert to upper case (if applicable)

```
#include <ctype.h>
#include <string.h>

long atox(char *s)
{
    char xdigs[] = "0123456789ABCDEF";
    long sum;

    /* Skip whitespace */
    while (isspace(*s))
        ++s;

    /* Do the conversion */
    for (sum = 0L; isxdigit(*s); ++s)
    {
        int digit = strchr(xdigs,toupper(*s)) - xdigs;
        sum = sum*16L + digit;
    }

    return sum;
}
```

<stdio.h>

- Formatted I/O
- Byte I/O
- String I/O
- Block I/O
- File Operations

Formatted I/O

Fixed-length argument lists

scanf	(standard input)
fscanf	(file input)
sscanf	(in-core input)
printf	(standard output)
fprintf	(file output)
sprintf	(in-core output)

Variable-length argument lists

vprintf	(standard output)
vfprintf	(file output)
vsprintf	(in-core output)

```c
#include <stdio.h>

long atox(char *s)
{
    long n;
    sscanf(s,"%x",&n);
    return n;
}

int main()
{
    int n = atox("1abf");
    printf("%x, %d\n",n,n);
    return 0;
}

/* Output:
1abf, 6847
*/
```

Byte I/O

getchar	(standard input)
getc	(file input)
ungetc	(affects file input)
fgetc	(file input)
putchar	(standard output)
putc	(file output)
fputc	(file output)

```
#include <stdio.h>

int copy(FILE *dest, FILE *source)
{
    int c;

    while ((c = getc(source)) != EOF)
        if (putc(c,dest) == EOF)
            return EOF;
    return 0;
}
```

String I/O

gets	(standard input)
fgets	(file input)
puts	(standard output)
fputs	(file output)

Block I/O

fread
fwrite

```
#include <stdio.h>

int copy(FILE *dest, FILE *source)
{
    size_t count;
    char buf[BUFSIZ];

    while (!feof(source))
    {
        count = fread(buf,1,BUFSIZ,source);
        if (ferror(source))
            return EOF;
        if (fwrite(buf,1,count,dest) != count)
            return EOF;
    }
    return 0;
}

int main()
{
     return copy(stdout, stdin);
}
```

File I/O

fopen
freopen
fclose
fflush

```
/* copy3.c */
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    if (argc == 3)
    {
        char s[BUFSIZ];
        FILE *inf, *outf;

        if ((inf = fopen(argv[1],"r")) == NULL)
            return EXIT_FAILURE;

        if ((outf = fopen(argv[2],"w")) == NULL)
            return EXIT_FAILURE;

        while (fgets(s,BUFSIZ,inf))
            fputs(s,outf);

        fclose(inf);
        fclose(outf);
        return EXIT_SUCCESS;
    }
    else
        return EXIT_FAILURE;
}
```

```
/* records.c: Illustrates file positioning */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAXRECS 10

struct record
{
    char last[16];
    char first[11];
    int age;
};

static char *get_field(char *, char *);

int main()
{
    int nrecs;
    char s[81];
    struct record recs[MAXRECS], recbuf;
    FILE *f;
```

```
/* Carefully store records */
for (nrecs = 0; nrecs < MAXRECS && get_field("Last",s);
     ++nrecs)
{
    strncpy(recs[nrecs].last,s,15)[15] = '\0';
    get_field("First",s);
    strncpy(recs[nrecs].first,s,10)[10] = '\0';
    get_field("Age",s);
    recs[nrecs].age = atoi(s);
}

/* Write records to file */
if ((f = fopen("recs.dat","w+b")) == NULL)
    return EXIT_FAILURE;
if (fwrite(recs,sizeof recs[0],nrecs,f) != nrecs)
    return EXIT_FAILURE;

/* Position at last record */
fseek(f,(nrecs-1)*sizeof(struct record),SEEK_SET);
fread(&recbuf,1,sizeof(struct record),f);
printf("last: %s, first: %s, age: %d\n",
  recbuf.last,recbuf.first,recbuf.age);
```

```
    /* Position at first record */
    rewind(f);
    fread(&recbuf,1,sizeof(struct record),f);
    printf("last: %s, first: %s, age: %d\n",
      recbuf.last,recbuf.first,recbuf.age);

    return EXIT_SUCCESS;
}

static char *get_field(char *prompt, char *buf)
{
    /* Prompt for input field */
    fprintf(stderr,"%s: ",prompt);
    return gets(buf);
}
```

File Operations

remove
rename

tmpfile
tmpnam

fgetpos
fsetpos
fseek
ftell
rewind

clearerr
feof
ferror

Other <stdio.h> stuff

Types

FILE	Encapsulates file access info
fpos_t	File Position (returned by fgetpos)

Macros

NULL	Zero pointer
EOF	Special value representing end-of-file
BUFSIZ	Preferred stream buffer size
FOPEN_MAX	Max # of files open simultaneously
FILENAME_MAX	Max # of characters in a file name minus 1
L_tmpnam	Max # of characters in a tempfile name minus 1
TMP_MAX	Max # of distinct filenames returned from tmpnam
SEEK_CUR	Signals fseek to seek relative to current position
SEEK_END	Signals fseek to seek from end-of-file
SEEK_SET	Signals fseek to seek from start-of-file

Walkthrough:
A File Viewing program

- view.c
- stack.h
- ansi.h

<stdlib.h>

- String Conversion
- Random Numbers
- Memory Management
- Environment Interface
- Searching and Sorting
- Integer Arithmetic
- Multibyte String Functions

String Conversion

- atoi ((int) strtol(s, NULL, 10))
- atol (strtol(s, NULL, 10))
- atof (strtod(s, NULL))
- strtod
- strtol
- stroul

```c
/* sconv.c */
#include <stdio.h>
#include <stdlib.h>

int main()
{
    const char* s = "  -123.45";
    int n = atoi(s);
    double x = atof(s);
    long m = strtol(s,NULL,8);
    printf("n == %d, x == %f, m == %ld\n",
           n, x, m);
    return 0;
}

/* Output:
n == -123, x == -123.450000, m == -83
*/
```

```c
/* strtol.c */
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char *input = "101 123 45678 90abc g";
    char *nextp = input;
    long bin, oct, dec, hex, beyond;

    bin = strtol(nextp,&nextp,2);
    oct = strtol(nextp,&nextp,8);
    dec = strtol(nextp,&nextp,10);
    hex = strtol(nextp,&nextp,16);
    beyond = strtol(nextp,&nextp,17);

    printf("bin = %ld\n",bin);
    printf("oct = %lo\n",oct);
    printf("dec = %ld\n",dec);
    printf("hex = %lx\n",hex);
    printf("beyond = %ld\n",beyond);
    return 0;
}

/* Output:
bin = 5
oct = 123
dec = 45678
hex = 90abc
beyond = 16
*/
```

A better atox( )

```
#include <stdlib.h>

long atox(char *s)
{
    return strtol(s,NULL,16);
}
```

Environment Interface

- getenv
- exit
- system

- abort
- atexit

Walkthrough:
An external sort/merge

- vsort.c
- Illustrates:
  - tmpnam
  - atexit
  - remove

Walkthrough:
Dealing Cards

- deal.c
- Illustrates:
  - random numbers
  - integer arithmetic (abs, div)

<string.h>

- Copying
- Concatenation
- Comparison
- Searching

Copying

- memcpy
- memmove
- strcpy
- strncpy

Concatenation

- strcat
- strncat

Comparison

- memcmp
- strcmp
- strncmp
- strcoll
- strxfrm

Searching

- memchr
- strchr
- strrchr
- strspn
- strcspn
- strstr
- strpbrk
- strtok

```c
/* find.c:    Extract lines from a file */
#include <stdio.h>
#include <string.h>

#define WIDTH 128

int main(int argc, char *argv[])
{
    char line[WIDTH];
    char *search_str;
    int lineno = 0;

    if (argc == 1)
        return 1;   /* Search string required */
    else
        search_str = argv[1];

    while (gets(line))
    {
        ++lineno;
        if (strstr(line,search_str))
            printf("%d:  %s\n",lineno,line);
    }

    return 0;
}
```

```
$ find str <find.c
```

```
4:  #include <string.h>
11:      char *search_str;
15:          return 1;   /* Search string required */
17:          search_str = argv[1];
22:          if (strstr(line,search_str))
```

Walkthrough
String Spanning

- span.c
- Illustrates:
  - strspn
  - strcspn
  - strpbrk

```
/* token.c: Splits stdin into tokens */
#include <stdio.h>
#include <string.h>

int main()
{
    char s[81];
    const char *break_set =
        " \t\n\\\"~!@#$%^&*()-_=+`'[]{}|;:/?.,<>";

    while (gets(s))
    {
        char *tokp, *sp = s;

        while ((tokp = strtok(sp,break_set)) !=
                NULL)
        {
            puts(tokp);
            sp = NULL;
        }
    }
    return 0;
}
```

Group II

Group II (tools for the professional)

<assert.h>	Defensive Programming
<limits.h>	Integral Parameters
<stddef.h>	Universal declarations
<time.h>	Date & Time Processing

Assertions

- Authoritative statements about code
  - design documentation in your code!
- For debugging an implementation
  - Not for error handling
- Doesn’t appear in released binaries

A Sample Assertion

```
#include <assert.h>
…

{
    long y;
    int m, d;
    j2g(jd, y, m, d);
    assert(m > 0);
    return daysToDate[isLeap(y)][m-1] + d;
}
```

A Sample Assertion

```
/* Make the 1's precede the 2's so that we get
 * non-negative answers */
if (order > 0)
{
    j2g(d.jd, y1, m1, d1);
    j2g(jd, y2, m2, d2);
}
else
{
    j2g(jd, y1, m1, d1);
    j2g(d.jd, y2, m2, d2);
}

long years = y2 - y1;
int months = m2 - m1;
int days = d2 - d1;
assert(years > 0 || years == 0 && months > 0 ||
       years == 0 && months == 0 && days > 0);
```

<stddef.h>

- ptrdiff_t
- size_t
- wchar_t
- NULL
- offsetof

```c
/* offsetof.c */
#include <stddef.h>
#include <stdio.h>

struct Person
{
    char name[15];
    int age;
};

int main()
{
    printf("%d\n",offsetof(struct Person, age));
    return 0;
}

/* Output:
16
[expected 15]
*/
```

<time.h>

- Stopwatch-like functions
- Current date and time
- Limited date arithmetic

Stopwatch Stuff

- clock( )
- clock_t
- CLOCKS_PER_SEC

```
/* timer.c: Stopwatch Functions */
#include <time.h>
#include "timer.h"

static clock_t start = (clock_t) 0;

/* Reset the timer */
void timer_reset(void)
{
    start = clock();
}

/* Wait a number of seconds */
void timer_wait(double secs)
{
    clock_t stop = clock() +
                   (clock_t) (secs * CLOCKS_PER_SEC);
    while (clock() < stop);
        ;
}

/* Compute elapsed time in seconds */
double timer_elapsed(void)
{
    return (double)(clock() - start) / CLOCKS_PER_SEC;
}
```

Time and Date Stuff

- time( )
- time_t
- localtime( )
- struct tm
- ctime( )
- mktime( )
- difftime( )
- strftime( )

Walkthrough
Formatting the Current Time

- tformat.c
- Illustrates:
  - time_t
  - time( )
  - localtime( )
  - strftime( )

Walthrough
Date Arithmetic

- Tmath.c
- Illustrates:
  - mktime
  - difftime

Group III

Group III (power at your fingertips when you need it)

<errno.h>	Error detection
<float.h>	Real arithmetic parms
<locale.h>	Cultural adaptation
<math.h>	Math functions
<setjmp.h>	Non-local branching
<signal.h>	Interrupt handling (sort of)
<stdarg.h>	Variable-length arg lists

Walkthrough
A Simple Calculator

- calc.c
- Illustrates:
  - errno
  - various math and string functions

Locales

- Cultural-specific info
  - monetary symbols, punctuation, dates
- setlocale( )
- Categories:
  - LC_COLLATE, LC_CTYPE, LC_MONETARY, LC_NUMERIC, LC_TIME
- Affects I/O functions

```c
/* locale.c: Doesn't work on all compilers */
#include <locale.h>
#include <stdio.h>
#include <time.h>

int main()
{
    char text[81];
    time_t timer = time(NULL);

    setlocale(LC_ALL,"french");
    printf("%f\n",1.2);
    puts(ctime(&timer));
    strftime(text, sizeof text,
             "%x %A %B %d\n",localtime(&timer));
    puts(text);
    return 0;
}

/* Output:
1,200000
Thu Jul 22 18:34:22 1999

22/07/99 jeudi juillet 22
*/
```

Walkthrough & Demo
Signal Handling

- ignore.c
- Usurps Control-C handling

Walkthrough
Non-local Branching

- setjmp.c
- Illustrates setjmp/longjmp

<stdarg.h>

- Allows you to build functions like printf
  - variable number of args
- You must pass information about the args to the function
  - number of args
  - format string
  - NULL terminator arg

Walkthrough
Finding a List Maximum

- max.c
- Illustrates:
  - va_list
  - va_start
  - va_arg
  - va_end

Walkthrough
Concatenating Multiple Strings

- concat.c

va_lists as Arguments

- vprintf/vsprintf/vfprintf
- Walkthrough fatal.c
- See Chapter 6