Ateneo de Manila University

The Standard C Library

Department of Information Systems and Computer Science
S.Y. 2001-2002
http://sysads.ateneo.net/wyu/
wyu@admu.edu.ph
Section I

Standard C Library
Standard C Library

★ Integer Functions
★ Random Numbers
★ String Conversion
★ Searching
★ Sorting
Integer Functions

- contains four basic integer functions

- `int abs(int number);`
  - functions return the absolute value of its number arguments

- `long int labs(long int number);`
  - long int version of the `abs()` function

- `div_t div(int numerator, int denominator);`
  - takes two arguments, numerator and denominator and produces a quotient and a remainder of the integer division
  - returns output in a `div_t` structure
  - `typedef struct {
      int quot;  /* quotient */
      int rem;  /* remainder */
  } div_t;`
ldiv_t ldiv(long int numerator, long int denominator);

- long int version of the *ldiv()* function
- returns output in a ldiv_t structure similar to the div_t structure
Random Numbers

- useful in programs that need to simulate random events, such as games, simulations and experimentations

- int rand(void);
  - returns successive pseudo-random numbers in the range from 0 to \((2^{15}) - 1\)

- void srand(unsigned int seed);
  - is used to set the seed
  - typically the time of the day is used here
  - srand( (unsigned int) time( NULL ));
★ generate pseudo-random numbers using the linear congruential algorithm and 48-bit integer arithmetic

★ \( X_{n+1} = (aX_n + c) \mod m \), where \( n \geq 0 \)

★ double drand48(void);
  – returns a double precision random number between 0.0 and 1.0

★ long lrand48(void);
  – returns a long integer precision random number between 0 and \( 2^{31} \)

★ long mrand48(void);
  – returns a long integer precision random number between \(-2^{31}\) and \(2^{31}\)

★ void srand48(long seed);
  – initialization function for seeding \( drand48() \), \( lrand48() \) and \( mrand48() \)
String Conversion

- double atof(char *string)
  - convert string to floating point value
- int atoi(char *string)
  - convert string to an integer value
- int atol(char *string)
  - convert string to a long integer value
- double strtod(char *string, char *endptr)
  - convert string to a floating point value
- long strtol(char *string, char *endptr, int radix)
  - convert string to a long integer using a given radix
- unsigned long strtoul(char *string, char *endptr, int radix)
  - convert string to unsigned long
Searching and Sorting

- two functions are provided for searching and sorting
  - void qsort(void *base, size_t num_elements, size_t element_size, int (*compare)(const void *, const void *));
  - void *bsearch(const void *key, const void *base, size_t nel, size_t size, int (*compare)(const void *, const void *));

- default structures for example

  typedef struct {
    int key;
    struct other_data;
  } Record;
☆ comparison function for example

```c
int record_compare(void const *a, void const *b)
{
    return ( ((Record *)a)->key -
            ((Record *)b)->key );
}
```

☆ to perform a binary search

```c
Record key;
Record *ans;
key.key = 3;
ans = bsearch(&key, array, arraylength,
              sizeof(Record), record_compare);
```

☆ to perform a quick sort

```c
qsort( array, arraylength, sizeof(Record),
      record_compare);
```
Section II

Mathematics
Math Library

★ contains a set of common mathematical functions
★ must be include \texttt{math.h} and link with \texttt{libm.a}

★ \texttt{double acos(double x)} - compute arc cosine of \(x\)
★ \texttt{double asin(double x)} - compute arc sine of \(x\)
★ \texttt{double atan(double x)} - compute arc tangent of \(x\)
★ \texttt{double atan2(double y, double x)} - compute arc tangent of \(y/x\)
★ \texttt{double ceil(double x)} - get smallest integral value that exceeds \(x\)
★ \texttt{double cos(double x)} - compute cosine of angle in radians
★ \texttt{double cosh(double x)} - compute the hyperbolic cosine of \(x\)
★ \texttt{double exp(double x)} - compute exponential of \(x\)
★ \texttt{double fabs (double x )} - compute absolute value of \(x\)
- double floor(double x) - get largest integral value less than x
- double fmod(double x, double y) - divide x by y with integral quotient and return remainder
- double frexp(double x, int *expptr) - breaks down x into mantissa and exponent of the number
- labs(long n) - find absolute value of long integer n
- double ldexp(double x, int exp) - reconstructs x out of mantissa and exponent of two
- double log(double x) - compute log(x)
- double log10 (double x) - compute log to the base 10 of x
- double modf(double x, double *intptr) - breaks x into fractional and integer parts
- double pow (double x, double y) - compute x raised to the power y
- double sin(double x) - compute sine of angle in radians
★ double sinh(double x) - compute the hyperbolic sine of x
★ double sqrt(double x) - compute the square root of x
★ double tan(double x) - compute tangent of angle in radians
★ double tanh(double x) - compute the hyperbolic tangent of x
Math Constants

★ **HUGE** - the maximum value of a single-precision floating-point number

★ **M_E** - the base of natural logarithms (e)

★ **M_LOG2E** - the base-2 logarithm of e

★ **M_LOG10E** - the base-10 logarithm of e

★ **M_LN2** - the natural logarithm of 2

★ **M_LN10** - the natural logarithm of 10

★ **M_PI** - \(\pi\)

★ **M_PI_2** - \(\frac{\pi}{2}\)

★ **M_PI_4** - \(\frac{\pi}{4}\)

★ **M_1_PI** - \(\frac{1}{\pi}\)

★ **M_2_PI** - \(\frac{2}{\pi}\)
- **M_2_SQRTP1** - \( \frac{2}{\sqrt{\pi}} \)
- **M_SQRT2** - \( \sqrt{2} \)
- **M_SQRT1_2** - \( \sqrt{\frac{1}{2}} \)
- **MAXFLOAT** - the maximum value of a non-infinite single-precision floating point number
- **HUGE_VAL** - positive infinity
Section III

Input/Output Functions
Reporting Errors

⋆ void perror(const char *message);
  – writes an error message to standard error
  – the error message is in the form of argument string message, then a colon and a blank, then the error message defined by the error return by the last system call and a newline

⋆ errno
  – special system constant defined in errno.h
  – used to represent a particular error
  – must declare extern int errno; in the program
Streams

★ is a portable way of writing data to different data sources
★ can be file or a physical device (e.g. printer or monitor) which is manipulated with a pointer to the stream
★ internal C data structure, FILE, which represents all streams and is defined in stdio.h
★ three basic input/output stream provided by a Unix system:
  - stdin - standard input (default keyboard)
  - stdout - standard output (default console)
  - stderr - standard error (default console)
★ redirection of these streams are possible:
  - program > file.out redirecting standard output to file.out
  - program < file.in redirecting standard input from file.in
- first | second redirecting the standard output of the first program to the standard input of the second program

Figure 1: Representation of Stream I/O Model
Common Stream I/O Operations

* `int fgetc(FILE *stream);`
  - read a single character from the file stream \( FILE \)

* `int getc(FILE *stream);`
  - equivalent to `fgetc` but implemented as a macro

* `int getchar(void);`
  - equivalent to `getc` on stdin

* `char *fgets(char *s, int size, FILE *stream);`
  - reads in at most one less than size characters from stream and stores them into the buffer pointed to by s and terminated by a '\0'
  - reading stops when a EOF or newline is encountered

* `char *gets(char *s);`
  - equivalent to `fgets` on stdin
∗ int ungetc(int c, FILE *stream);
   – pushes character c back to the stream FILE
Formatted Output

⋆ family of function used to produce particular output according to a format defined by a format string

⋆ int printf(const char *format, ...);
  – displays formatted output for stdout

⋆ int fprintf(FILE *stream, const char *format, ...);
  – writes formatted output to a stream defined by FILE

⋆ int sprintf(char *str, const char *format, ...);
  – writes formatted output to a string

⋆ int snprintf(char *str, size_t size, const char *format, ...);
- writes only \texttt{size} bytes output into a string

- \texttt{\%d} - represents an integer value
- \texttt{\%x} - represents an integer value to be displayed in hexadecimal notation
- \texttt{\%3.2f} - represents a floating point number with 3 digits and 2 digits for its decimal part
- \texttt{\%e} - represents a number in scientific notation
- \texttt{\%c} - represents a character
- \texttt{\%5s} - represents a string of 5 characters
Formatted Input

* family of function used to accept a particular input according to a format defined by a format string

* `int scanf( const char *format, ...);`
  - input a value based on the format string from `stdin`

* `int fscanf( FILE *stream, const char *format, ...);`
  - input a value based on the format string from a stream defined by `FILE`

* `int sscanf( const char *str, const char *format, ...);`
  - input a value based on the format string from a string

* conversion specifiers for formatted output apply to formatted input functions
Manipulating Streams

- standard streams such as stdin, stdout and stderr need not be opened and closed

- `FILE *fopen (const char *path, const char *mode);`
  - opens the file whose name is the string pointed to by path and associates a stream with it
  - the modes in which the stream will be accessed is defined by:
    - `r` Open text file for reading. The stream is positioned at the beginning of the file.
    - `r+` Open for reading and writing. The stream is positioned at the beginning of the file.
    - `w` Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.
    - `w+` Open for reading and writing. The file is created if it does not exist,
otherwise it is truncated. The stream is positioned at the beginning of the file.

a Open for writing. The file is created if it does not exist. The stream is positioned at the end of the file.

a+ Open for reading and writing. The file is created if it does not exist. The stream is positioned at the end of the file.

★ int fflush(FILE *stream);
   – forces a write of all user-space buffered data for the given output or update stream via the stream’s underlying write function
   – stream argument is NULL, fflush flushes all open output streams

★ int fclose(FILE *stream);
   – writes current buffer and then closes the stream
   – dissociates the named stream from its underlying file or set of functions

★ int feof(FILE *stream);
   – returns a value greater than 0 if the stream is at EOF
star int ferror(FILE *stream);
  – reports on the error state of the stream and returns a value greater than zero if an error has occurred

star void clearerr(FILE *stream);
  – resets the error indication for a given stream

star int fileno(FILE *stream);
  – returns the integer file descriptor associated with the named stream