C Library Calls

An Advanced Introduction to Unix/C Programming

John Dempsey
COMP-232 Programming Languages
California State University, Channel Islands

C Library Function Calls

- Library calls make it easier to write programs by providing functions you can call.
- Library calls are defined in Section 3 of the manual pages.
- To call a library function, an include file is necessary.

/usr/include Header Files

- There are a large number of ANSI Standard Libraries defined. Function prototypes can be found in the following /usr/include files:

assert.h	signal.h
ctype.h	stdarg.h
errno.h	stddef.h
float.h	stdio.h
limits.h	stdlib.h
locale.h	string.h
math.h	time.h
setjmp.h	(There are ~115 .h files in /usr/include)

Where's the object code for stdio.h?

#include <stdio.h> defines the interface/prototypes in /usr/include/stdio.h.

But where is the compiled object code for, say, the fprintf function defined in stdio.h?

```
#include <stdio.h>
fprintf(stdout, "%s, %s %d, %.2d:%.2d\n",
        weekday, month, day, hour, min);
```

/usr/lib/x86_64-linux-gnu for Ubuntu

```
john@oho:/usr/lib/x86_64-linux-gnu$ ls *.a
libBrokenLocale.a  libc_nonshared.a  libdl.a  libl.a       libmcheck.a       libmysqlservices.a  libresolv.a  libssl.a
libanl.a           libcrypt.a        libfl.a  libm-2.31.a  libmvec.a         libnsl.a            librpcsvc.a  libutil.a
libc.a             libcrypto.a       libg.a   libm.a       libmysqlclient.a  libpthread.a        librt.a      libz.a

john@oho:/usr/lib/x86_64-linux-gnu$ ar tvf libc.a|grep printf
rw-r--r-- 0/0   1392 Dec 31 16:00 1969 vfprintf.o
rw-r--r-- 0/0   1416 Dec 31 16:00 1969 vprintf.o
rw-r--r-- 0/0  20296 Dec 31 16:00 1969 printf_fp.o
rw-r--r-- 0/0   3104 Dec 31 16:00 1969 reg-printf.o
rw-r--r-- 0/0   1680 Dec 31 16:00 1969 fprintf.o          ← fprintf object code
rw-r--r-- 0/0   1752 Dec 31 16:00 1969 printf.o           ← printf object code

john@oho:/usr/lib/x86_64-linux-gnu$ ar tvf libc.a|grep assert
rw-r--r-- 0/0   3984 Dec 31 16:00 1969 assert.o
```

assert.h

assert.h Library Functions		
Function	**Function Prototype**	**Description**
assert	void assert(scalar expression);	If expression is false (i.e., compares equal to zero), assert prints an error message to standard error and terminates the program by calling abort().

man assert

```
john@oho:~$ man assert | cat
ASSERT(3)                  Linux Programmer's Manual                  ASSERT(3)

NAME
       assert - abort the program if assertion is false

SYNOPSIS
       #include <assert.h>

       void assert(scalar expression);

DESCRIPTION
       This  macro  can help programmers find bugs in their programs, or handle exceptional cases via a crash that will produce lim-
       ited debugging output.

       If expression is false (i.e., compares equal to zero), assert() prints an error message to standard error and terminates  the
       program  by calling abort(3).  The error message includes the name of the file and function containing the assert() call, the
       source code line number of the call, and the text of the argument.

RETURN VALUE
       No value is returned.

SEE ALSO
       abort(3), assert_perror(3), exit(3)
```

assert

```
john@oho:~$ cat assert.c
#include <stdio.h>
#include <assert.h>
void main()
{
        int     i = 1;
        int     j = 2;

        assert(i == 1);
        printf("i = 1\n");
        assert(i == j);
        printf("i = j\n");
        printf("End of program\n");
}
john@oho:~$ gcc assert.c; a.out
i = 1
a.out: assert.c:12: main: Assertion `i == j' failed.
Aborted (core dumped)
```

ctype.h

ctype.h Library Functions		
Function	**Function Prototype**	**Description**
isalnum	int isalnum(int c);	Checks for alphanumeric character.
isalpha	int isalpha(int c);	Checks for alphabetic character.
isascii	int isascii(int c);	Checks whether c is a 7-bit unsigned character that fits into the ASCII character set.
isblank	int isblank(int c);	Checks for a space or tab character.
iscntrl	int iscntrl(int c);	Check for a control character.
isdigit	int isdigit(int c);	Check for a digit (0 through 9).
isgraph	int isgraph(int c);	Check for any printable character except space.
islower	int islower(int c);	Checks for a lowercase character.
isprint	int isprint(int c);	Checks for printable character including space.
ispunct	int ispunct(int c);	Checks for punctuation character not space.
isspace	int isspace(int c);	Checks for space, \f, \n, \r, \t, or \v characters.

Why an int versus a char?

In ISO/IEC 9899:1999 (the old C standard), it says:

§7.4 Character handling

The header declares several functions useful for classifying and mapping characters. In all cases the argument is an int, the value of which shall be representable as an unsigned char or shall equal the value of the macro EOF. If the argument has any other value, the behavior is undefined.

(*I've left out a footnote.*) Both C89 and C11 say very much the same thing.

As long as c is in the range of integers that an unsigned char can store (and there are 8 bits per character, EOF is -1, and the initialization is correct), then this works beautifully.

ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

isdigit() will check to see if the int is in the decimal range of 48 to 57.

isalpha() will check to see if the int is in the decimal range of 65 to 90 or 97 to 122.

ctype.h

ctype.h Library Functions		
Function	**Function Prototype**	**Description**
isupper	int isupper(int c);	Checks for uppercase
isxdigit	int isxdifit(int c);	Checks for hexadecimal digits (0 through F).
tolower	int tolower(int c);	Returns lowercase equivalent.
toupper	int toupper(int c);	Returns uppercase equivalent.

ctype.h

```
john@oho:~$ cat ctype.c
#include <stdio.h>
#include <ctype.h>
void main() {
        char    string[20] = "Aa8# ";
        if (isalnum(string[0])) printf("%c is an alphanumeric character.\n",
            string[0]);
        if (isblank(string[4])) printf("%c is a blank.\n", string[4]);
        if (isdigit(string[2])) printf("%c is a digit.\n", string[2]);
        if (islower(string[1])) printf("%c is a lower case character.\n", string[1]);
        if (ispunct(string[3])) printf("%c is a punctuation character.\n", string[3]);
        if (isxdigit(string[0])) printf("%c is a hexadecimal digit.\n", string[0]);
        if (isupper(string[0])) printf("%c is now lower case.\n", tolower(string[0]));
        if (!islower(string[0])) printf("%c is not lower case.\n", string[0]);
        if (iscntrl(string[5])) printf("%02x is a control character.\n", string[5]);
}
```

```
john@oho:~$ gcc ctype.c; a.out
A is an alphanumeric character.
  is a blank.
8 is a digit.
a is a lower case character.
# is a punctuation character.
A is a hexadecimal digit.
a is now lower case.
A is not lower case.
07 is a control character.
```

math.h

math.h Library Functions		
Function	**Function Prototype**	**Description**
acos	double acos(double x);	Arc cosine function.
asin	double asin(double x);	Arc sine function.
atan	double atan(double x);	Arc tangent function.
atan2	double atan2(double y, double x);	Arch tangent function of two variables.
atof	double atof(const char *nptr);	Convert string to a double.
ceil	double ceil(double x);	Returns the smallest integral value not less than x.
cos	double cos(double x);	Cosine function.
cosh	double cosh(double x);	Hyperbolic cosine function.
exp	double exp(double x);	The value of e raised to the power of x.
fabs	double fabs(double x);	Absolute value of the floating point number x.
floor	double floor(double x);	Largest integral value not greater than x.

math.h

math.h Library Functions		
Function	**Function Prototype**	**Description**
fmod	double fmod(double x, double y);	Compute floating-point remainder of x/y.
frexp	double frexp(double x, int *exp);	Splits number x into normalized fraction and exponent stored in exp.
ldexp	double ldexp(double x, int exp);	Multiplies floating-point number x by 2 raised to the power of exp.
log	double log(double x);	Returns natural logarithm of x.
log10	double log10(double x);	Returns the base 10 logarithm of x.
pow	double pow(double x, double y);	Returns the value of x raised to the power of y.
sin	double sin(double x);	Return sine of x, where x is given in radians.
sinh	double sinh(double x);	Return hyperbolic sine of x.
sqrt	double sqrt(double x);	Returns square root of x.
tan	double tan(double x);	Return tangent of x, where x is given in radians.
tanh	double tanh(double x);	Returns hyperbolic tangent of x.

pwd.h

pwd.h Library Functions		
Function	**Function Prototype**	**Description**
getpwnam	struct passwd *getpwnam(const char *name);	Returns passwd structure contents for name.

The passwd structure is defined in <pwd.h> as follows:

```
struct passwd {
    char   *pw_name;        /* username */
    char   *pw_passwd;      /* user password */
    uid_t   pw_uid;         /* user ID */
    gid_t   pw_gid;         /* group ID */
    char   *pw_gecos;       /* user information */
    char   *pw_dir;         /* home directory */
    char   *pw_shell;       /* shell program */
};
```

stdarg.h

stdarg.h Library Functions		
Function	**Function Prototype**	**Description**
va_arg	type va_arg(va_list ap, type);	Expands to an expression that has the type and value of the next argument in the call.
va_end	void va_end(va_list ap);	Each invocation of va_start must be matched by a cooresponding invocation of va_end.
va_start	void va_start(va_list ap, last);	Initializes ap for subsequent use by va_arg and va_end, and must be called first. last is the name of the last argument before the variable argument list.

A function may be called with a varying number of arguments of varying types. The include file stdarg.h declares a type va_list and defines three macros for stepping through a list of arguments whose number and types are not known to the called function.

The called function must declare an object of type va_list which is used by va_start, va_arg, and va_end.

stdarg.h

```
#include<stdio.h>
#include<stdarg.h>

void add_values(int count, ...)
{
   int       i;
   int       sum = 0;
   int       value;
   va_list    vlist;

   va_start(vlist, count);
   for(i=0; i<count; ++i) {
     value = va_arg(vlist, int);
     sum = sum + value;
     if (i+1 == count)
       printf("%d", value);
     else
       printf("%d + ", value);
   }
   printf(" = %d\n", sum);
   va_end(vlist);
}
```

```
int main()
{
   add_values(3, 1, 2, 3);
   add_values(5, 10, 20, 30, 40, 50);
   return 0;
}

john@oho:~$ gcc stdarg.c; a.out
1 + 2 + 3 = 6
10 + 20 + 30 + 40 + 50 = 150
```

stdlib.h

stdlib.h Library Functions		
Function	**Function Prototype**	**Description**
free	void free(void *ptr);	Frees memory allocated by malloc, calloc, or realloc pointed to by ptr.
freopen	FILE *freopen(const char *pathname, const char *mode, FILE *stream);	Opens file named by pathname and associates stream pointed to by stream. Closes existing stream.
ldiv	ldiv_t ldiv(long numerator, long denominator);	Computes numerator/denominator.
malloc	void *malloc(size_t size);	Allocates size bytes and returns pointer to memory. Memory is not initialized.
mblen	int mblen(const char *s, size_t n);	Inspects at most n bytes starting at s and extracts the next complete multibyte character.
mbtowc	int mbtowc(wchar_t *pwc, const char *s, size_t n);	Convert multibyte sequence to a wide character.
qsort	void qsort(void *base, size_t nmemb, size_t size, int (*compar)(const void *, const void);	Sorts an array with nmemb elements of size size. base points to the start of the array.
rand	int rand(void);	Pseudo-random number generator.

stdlib.h

stdlib.h Library Functions		
Function	**Function Prototype**	**Description**
realloc	void *realloc(void *ptr, size_t size);	Changes size of memory block pointed to by ptr to size bytes.
strtol	long double strtol(const char *nptr, char **endptr, int base);	Converts string in nptr to a long integer according to the given base.
strtoul	unsigned long int strtoul(const char *nptr, char **endptr, int base);	Converts string in nptr to a long integer according to the given base which must be between 2 and 36.
system	int system(const char *command);	Runs command by creating a child process.
wctomb	int wctomb(char *s, wchar_t wc);	Convert a wide character to a multibyte sequence.

stdlib.h - system

```
john@oho:~$ cat system.c
#include <stdio.h>
#include <stdlib.h>

void main()
{
        system("echo ---;mount | grep C; echo ---;df -k /home; echo ---; uptime; echo ---;");
}

john@oho:~$ gcc system.c; a.out
---
C:\ on /mnt/c type drvfs (rw,noatime,uid=1000,gid=1000,case=off)
---
Filesystem      1K-blocks       Used Available Use% Mounted on
rootfs        1951597728 1251181988 700415740  65% /
---
 16:22:05 up  6:55,  0 users,  load average: 0.52, 0.58, 0.59
---
```

stdlib.h – calloc, free

```
john@oho:~$ cat calloc.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX_NAMES       10
void main() {
        typedef struct name {
                char    first_name[15];
                char    last_name[20];
                int     age;
        } NAME;

        NAME *base_ptr;
        NAME *ptr;

        base_ptr = calloc(MAX_NAMES, sizeof(NAME));
        ptr = base_ptr;

        strcpy(ptr->first_name, "Tom");
        strcpy(ptr->last_name, "Smith");
        ptr->age = 19;
        printf("ptr->first_name=%s, last_name=%s, age=%d\n",
                ptr->first_name, ptr->last_name, ptr->age);

        ptr = ptr + 1;
        strcpy(ptr->first_name, "Suzy");
        strcpy(ptr->last_name, "Landwer");
        ptr->age = 24;

        printf("ptr+1->first_name=%s, last_name=%s, age=%d\n",
                ptr->first_name, ptr->last_name, ptr->age);

        free(base_ptr);
}


john@oho:~$ gcc calloc.c; a.out
ptr->first_name=Tom, last_name=Smith, age=19
ptr+1->first_name=Suzy, last_name=Landwer, age=24
```

stdlib.h – malloc, free

```
john@oho:~$ cat malloc.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX_NAMES       10
void main() {
        typedef struct name {
                char    first_name[15];
                char    last_name[20];
                int     age;
        } NAME;

        NAME *base_ptr;
        NAME *ptr;

        base_ptr = malloc(sizeof(NAME)*MAX_NAMES);
        ptr = base_ptr;

        strcpy(ptr->first_name, "Tom");
        strcpy(ptr->last_name, "Smith");
        ptr->age = 19;
        printf("ptr->first_name=%s, last_name=%s, age=%d\n",
                ptr->first_name, ptr->last_name, ptr->age);

        ptr = ptr + 1;
        strcpy(ptr->first_name, "Suzy");
        strcpy(ptr->last_name, "Landwer");
        ptr->age = 24;

        printf("ptr+1->first_name=%s, last_name=%s, age=%d\n",
                ptr->first_name, ptr->last_name, ptr->age);

        free(base_ptr);
}


john@oho:~$ gcc malloc.c; a.out
ptr->first_name=Tom, last_name=Smith, age=19
ptr+1->first_name=Suzy, last_name=Landwer, age=24
```

stdlib.h – atoi, atof, atoll, rand

```
john@oho:~$ cat atoi.c
#include <stdio.h>
#include <stdlib.h>

void main()
{
        int     i = 0;
        float   f = 0.0;
        long    l = 0;
        int     random_number = 0;

        i = atoi("1234");
        f = atof("5.678");
        l = atol("12345678909876543321");

        printf("i = %d\n",  i);
        printf("f = %f\n",  f);
        printf("l = %ld\n", l);
```

```
 random_number = rand();
        printf("random_number #1 = %d\n", random_number);
        random_number = rand();
        printf("random_number #2 = %d\n", random_number);
}


john@oho:~$ gcc atoi.c; a.out
i = 1234
f = 5.678000
l = 1234567890987654321
random_number #1 = 1804289383
random_number #2 = 846930886
```

string.h

string.h Library Functions		
Function	**Function Prototype**	**Description**
memchr	void *memchr(const void *s, int c, size_t n);	Scans initial n bytes of memory area pointed to by s for the first instance of c.
memcmp	int memcmp(const void *s1, const void *s2, size_t n);	Compares first n bytes of memory areas s1 and s2.
memcpy	void *memcpy(void *dest, const void *src, size_t n);	Copies n bytes from memory area src to memory area dest.
memmove	void *memmove(void *dest, const void *src, size_t n);	Copies n bytes from memory area src to memory area dest.
memset	void *memset(void *s, int c, size_t n);	Fills the first n bytes of memory pointed to by s with the constant byte c.
strcat	char *strcat(char *dest, const char *src);	Appends string src to the dest string.
strchr	char *strchr(const char *s, int c);	Returns a pointer to first occurrence of the character c in the string s or NULL if not found.
strcmp	int strcmp(const char *s1, const char *s2);	Compares the two strings s1 and s2.

string.h

string.h Library Functions		
Function	**Function Prototype**	**Description**
strcpy	char *strcpy(char *dest, const char *src);	Copy a string pointed to by src to dest.
strspn	size_t strspn(const char *s, const char *accept);	Calculates the length in bytes of the initial segment of s which consists entirely of bytes in accept.
strcspn	size_t strcspn(const char *s, const char *reject);	Calculates the length of the initial segment of s which consists entirely of bytes not in reject.
strdup	char *strdup(const char *s);	Returns pointer to a new string which is a duplicate of string s.
strerror	char *strerror(int errnum);	Returns a string that describes the error code passwd in the errnum argument.
strlen	size_t strlen(const char *s);	Calculates the length of the string pointed to by s, excluding the terminating null byte \0.
strncat	char *strncat(char *dest, const char *src, size_t n);	Appends at most n bytes from src to dest.
strncmp	int strncmp(const char *s1, const char *s2, size_t n);	Compares only the first n bytes of s1 and s2.

string.h

string.h Library Functions		
Function	**Function Prototype**	**Description**
strncpy	char *strncpy(char *dest, const char *src, size_t n);	Copies n bytes from src to dest.
strpbrk	char *strpbrk(const char *s, const char *accept);	Locates the first occurrence in string s of any of the bytes in string accept.
strrchr	char *strrchr(const char *s, int c);	Returns pointer to the last occurrence of the character c in string s.
strspn	size_t strspn(const char *s, const char *accept);	Calculates the length in bytes of the initial segment of s which consist so entirely of bytes in accept.
strstr	char *strstr(const char *haystack, const char *needle);	Finds the first occurrence of substring needle in string haystack, or NULL if not found.
strtok	char *strtok(char *str, const char *delim);	Divides a string into a sequence of zero or more tokens. First call str is set. NULL otherwise.

string.h – strcpy, strlen, strcat, memcpy, strncmp, strstr

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

void main()
{
      int    length;
      char   *ptr;
      char    string[100];
      char    string2[100];

      strcpy(string, "An apple a day");
      length = strlen(string);
      printf("string=:%s:, length=%d\n", string, length);

      strcat(string, " keeps the doctor away.");
      length = strlen(string);
      printf("string=:%s:, length=%d\n", string, length);
```

```
 memcpy(string2, string, 8);
      if (strncmp(string, string2, 6) == 0) {
        ptr = strncpy(string2, string, 6);
        printf("string2 = :%s:\n", string2);
        printf("ptr = :%s:\n", ptr);
        printf("The first 6 characters in string and string2 are the same.\n");
      }

      if ((ptr = strstr(string, "doctor")) != 0)
        printf("doctor is found in string starting here: %s\n", ptr);
}

john@oho:~$ gcc string.c; a.out
string=:An apple a day:, length=14
string=:An apple a day keeps the doctor away.:, length=37
string2 = :An apple:
ptr = :An apple:
The first 6 characters in string and string2 are the same.
doctor is found in string starting here: doctor away.
```

string.h - strtok

```
john@oho:~$ cat strtok.c
#include <stdio.h>
#include <string.h>

void main()
{
      char    string[100];
      char    *token;

      strcpy(string, "john:x:1000:1000:John Dempsey:/home/john:/bin/bash");

      token = strtok(string, ":");

      while (token != NULL) {
        printf("token = %s\n",  token);
        token = strtok(NULL, ":");
      }
}
```

```
john@oho:~$ gcc strtok.c; a.out
token = john
token = x
token = 1000
token = 1000
token = John Dempsey
token = /home/john
token = /bin/bash
```

time.h

time.h Library Functions		
Function	**Function Prototype**	**Description**
asctime	char *asctime(const struct tm *tm);	Initializes the tm structure found in time.h.
clock	clock_t clock(void);	Returns approximation of processor time used by program.
ctime	char *ctime(const time_t *timep);	Equivalent to asctime(localtime(t)).
difftime	double difftime(time_t time1, time_t time0);	Calculate the time difference in seconds.
gmtime	struct tm *gmtime(const time_t *timep);	Converts timep to Universal Time (UTC).
localtime	struct tm *localtime(const time_t *timep);	Converts timep to user specified timezone.
sleep	unsigned int sleep(unsigned int seconds);	Sleep for the number of seconds or until asignal arrives which is not ignored.
strftime	size_t strftime(char *s, size_t max, const char *format, const struct tm *tm);	Formats the time in tm according to format and places result into s of size max.

time.h - struct tm

```
struct tm {
        int tm_sec;         /* Seconds (0-60) */
        int tm_min;         /* Minutes (0-59) */
        int tm_hour;        /* Hours (0-23) */
        int tm_mday;        /* Day of the month (1-31) */
        int tm_mon;         /* Month (0-11) */
        int tm_year;        /* Year - 1900 */
        int tm_wday;        /* Day of the week (0-6, Sunday = 0) */
        int tm_yday;        /* Day in the year (0-365, 1 Jan = 0) */
        int tm_isdst;       /* Daylight saving time */
};
```

time.h – asctime, ctime, gmtime, localtime, time

```
#include <stdio.h>
#include <time.h>

void main()
{
      struct tm  *ptr;
      long       time_value;

      time(&time_value);
      printf("ctime          = %s\n", ctime(&time_value));

      ptr = localtime(&time_value);
      printf("asctime        = %s\n", asctime(ptr));

      printf("Local Date/Time = %02d/%02d/%04d %02d:%02d:%02d\n\n",
            ptr->tm_mon+1, ptr->tm_mday, ptr->tm_year+1900,
            ptr->tm_hour, ptr->tm_min, ptr->tm_sec);

      ptr = gmtime(&time_value);
      printf("GMT Date/Time = %02d/%02d/%04d %02d:%02d:%02d\n",
            ptr->tm_mon+1, ptr->tm_mday, ptr->tm_year+1900,
            ptr->tm_hour, ptr->tm_min, ptr->tm_sec);
}
```

```
john@oho:~$ gcc time.c; a.out
ctime          = Tue Feb  1 11:41:31 2022

asctime        = Tue Feb  1 11:41:31 2022

Local Date/Time  = 02/01/2022 11:41:31

GMT Date/Time   = 02/01/2022 19:41:31
```