Library and function of C

Dr. Donald Davendra *Ph.D.*

Department of Computing Science, FEI VSB-TU Ostrava

Description of functions and macros and their standard libraries

<assert.h> Macro used for debugging.
<ctype.h> Working with characters.

isalnum()	digit or lowercase or uppercase
isalpha ()	No. small and large letters
iscntrl ()	control characters
isdigit ()	digit
isgraph ()	characters with graphic design
islower ()	lower case
isprint ()	printable characters
ispunct ()	punctuation characters
isspace ()	whitespace
isupper ()	upper case
isxdigit ()	hexadecimal digits
tolower()	converting uppercase letters to lowercase
toupper()	converting lowercase letters to uppercase

Math Libraries

<math.h>

sin (x)	$\sin(x)$
cos(x)	$\cos(x)$
tan(x)	$\tan(x)$
asin (x)	$\arcsin(x)$
acos(x)	$\arccos(x)$
atan(x)	$\arctan(x)$
atan2(x,y)	arguments x, y
sinh (x)	$\sinh(x)$
cosh(x)	$\cosh(x)$
tanh(x)	$\tanh(x)$
exp(x)	e^x
log (x)	$\ln(x)$
log10(x)	$\log(x)$
pow(x,y)	x^y
sqrt (x)	$\sqrt{}(x)$
ceil (x)	$y \in Z, y \geq x$, lower
floor (x)	$y \in Z, y \leq x$, upper
fabs((x)	$\|x\|$
ldexp(x,n)	$x \times 2^n$
frexp(x, exp)	$x = m \times 2^exp$
modf(x, n)	x and n to the power of $+$
fmod(x,n)	Real remainder after dividing x/y

Error

1. Mathematical functions arguments and return valuesare of type double.
2. If an error the output is errno
   EDOM - outside the domain of input functions.
   ERANGE - Output HUGE_VAL or 0.0.

```
int i
double g, f;

printf("%f \n" , ceil (3.14)); 4.000000
printf("%f \n" , floor (3.14)); 3.000000
f = frexp(3.14, &g);

printf("%f x2 %d\n", f, i); 0.785000x2 2
f = modf(  3.14, &g)

printf("%f + %f\n" , g, f );    3.000000
printf('%f \n" , fmod(3.14 , 2.0)); 1.140000
```

<locale.h>Customizing the C locale.
Problem

- character set, such as isupper (W)
- decimal point versus decimal comma ours,
- write the time and date, eg 10.6.1995 or 6-10-1995 or July 10, 1995.

setlocale () adjusted
localeconv() returns

<setjmp.h> Allowing long jumps.

setjmp() prepare conditions for the return point

longjmp() performs non-local jump.

setjmp() saves and longjmp() restores the contents of registers required processor type is jmp_buf.

The principle of uniform response to the error with a subsequent restart of the program.

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

void chyba(void ); /     funkcni prototyp     /
jmp buf hlavni smycka ; /     globalni promenna /

int main(void)
{
        int i,j;

        printf ("Zacatek programu \n");
        hlavni smycka = setjmp ( ) ;

        do {
                printf ( Zadej dve cisla :    );
                scanf(   %d%d   ,&i ,&j); if(j !=0)
                printf(   %d/%d=%d\n   ,i,j, i/j); else
                chyba ( ) ;
        } while(i !=0);
}
```

```
void chyba(void)
{
        printf ("Chyba     nulovy delitel \n\n");
        longjmp ( hlavni smycka , 1 ) ;
}


void chyba(jmp buf kam se vratit , char  chybov)
{
        printf ("Chyba: %s\n" , chybova zprava );
        longjmp(kam se vratit , 1);
}
```

<signal.h> Signal Processing

signal () new signal

raise () transmitting the signal

```
#include <stdio .h>
#include <signal .h>
#include <setjmp .h>
#define POCET POKUSU 2
#define vynech 5000
/    znak se vypisuje jednou za vynech /

jmp buf navrat ;
char vypisovany znak =     #     ;

void obsluha signalu ( int nevyuzity parametr )
{static int pocet vstupu = 0;
        pocet vstupu++;
        if ( pocet vstupu <= pocet pokusu ) {
                printf ("\n%d:A zrovna ne \n" , pocet vstupu );
                vypisovany znak++; }
                longjmp ( navrat , pocet vstupu ) ;
}
```

```
int main(void)
{
        long int i = 0;
        int pid;
        void (     stara adr )( int );
        /      zavedeni nove adresy obsluhy SIGQUIT      /
        stara      adr = signal (SIGQUIT , obsluha      signalu ) ;
        /     ziskani identifikacniho cisla procesu     /
        pid=getpid ();
        printf("\nPID=%d \n\n" , pid);
        printf("Zadej trikrat prikaz: kill %d%d \n", SIGQUIT, pid);

        if (setjmp(navrat) < pocet pokusu) {
                while (1) {
                        if (++i % vynech == 0) {
                                putchar(vypisovany znak);
                                fflush(stdout); } } }
        printf("\n Tak tedy ano ...\n");
        /      obnoveni puvodni adresy obsluhy SIGQUIT      /
        signal (SIGQUIT , stara      adr ) ;
}
```

stdarg.h

<stdarg.h> Working with a variable number of parameters.

va_start ()	sets the beginning
va_arg()	gives another list item
va_end()	ends with a list of work

<stdio.h> functions for input and output.

stdio.h

<stdio.h> - input and output

fopen()	Opens a file
freopen()	Reopen stream with different file or mode
fflush ()	Flush stream
fclose ()	Close file
remove()	Remove file
rename()	Rename file
tmpfile ()	Open a temporary file
tmpnam()	Generate temporary filename
setbuf ()	Set stream buffer
setvbuf ()	Change stream buffering
fprintf ()	Write formatted data to stream
printf ()	Print formatted data to stdout
sprintf ()	Write formatted data to string
vfprintf ()	Write formatted data from variable argument list to stream

stdio.h

vprintf ()	Print formatted data from variable argument list to stdout
vsprintf ()	Write formatted data from variable argument list to string
fscanf ()	Read formatted data from stream
scanf ()	Read formatted data from stdin
sscanf ()	Read formatted data from string
vfscanf ()	Read formatted data from file into variable argument list
vsscanf ()	Read formatted data from string into variable argument list
fgetc ()	Get character from stream
getc ()	Get character from stream
fputs ()	Write string to stream
puts ()	Write string to stdout
getchar ()	Get character from stdin

stdlib.h

<stdio.h> Function for file streams

fread ()	Read block of data from file stream
fwrite ()	Write block of data to file stream
fseek ()	Reposition stream position indicator
ftell ()	Get current position in file stream
rewind()	Set position of file stream to the beginning
fgetpos()	Get current position in file stream
fsetpos()	Set position indicator of file stream
clearerr ()	Clear error indicators
feof ()	Check end-of-file indicator
ferror ()	Check error indicator
perror ()	Print error message

stdlib.h

<stdlib.h> Environment functions.

atof()	Convert string to double
atoi()	Convert string to integer
atol()	Convert string to long integer
strtod()	Convert string to double
strtol()	Convert string to long integer
strtoul()	Convert string to unsigned long integer
rand()	Generate random number
srand()	Initialize random number generator
calloc()	Allocate space for array in memory
malloc()	Allocate memory block
realloc()	Reallocate memory block
free()	Deallocate space in memory

stdlib.h

<stdlib.h> Environment functions.

abort()	Abort current process
exit ()	Terminate calling process
atexit ()	Set function to be executed on exit
system()	Execute system command
getnev()	Get environment string
bsearch()	Binary search in array
qsort ()	Sort elements of array
abs()	Absolute value
labs ()	Absolute value of type long int
div(x, y)	x/y a $x\%y$ division of type int
ldiv (x, y)	x/y a $x\%y$ division of type long int

Converting string to number

NOTE: ISO C does not define the reverse function, i.e. to convert a number to strings, eg itoa(). In these case it becomes necessary to use function sprintf().

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

int main(int argc, char *argv[]) {

char *s, char *chyba;
long int l;

s = "1234567";
l = atol(s);

printf("s=%s l =%ld\n",s,l); // s=1234567 l =1234567
s="1234567";

l= strtol(s, &chyba , 10);
printf("s=%s l =%ld\n",s,l); // s=1234567 l =1234567

s="ffff";
l=strtol (s, &chyba ,16);
printf("s=%s l=%ld \n",s,l); // s=ffff l=65535

s= "0xFGFFF";
l=strtol (s, &chyba , 16);
printf("s=%s l=%ld \nchybny znak:%s \n",s, l ,chyba);  //s=0xFGFFF l=15

s= 1111111111111111;
```

Generating Pseudorandom Number

1. int rand(void)
   Range $< 0, RAND\_MAX$), where the values betweena acertain range can be obatined as.
   Example: rand() % number, or rand() % 20

   #define real_rand()
   ((double)rand()/(RAND_MAX+1.0))
2. void srand(unsigned int start)
   Initialize the random number generator rand() using the seed start.

Generating Pseudorandom Number

```
#include <iostream>
#include <ctime>
#include <stdio.h>


int main(int argc, char *argv[]) {

        int i;

        srand ( ( unsigned ) time (NULL ) ) ;
        printf ("\nA range of numbers from 0 to 99\n");

        for( i=0; i<10; i++)
                printf("%2d " , rand() % 100);

        return 0;
}
```

Function for dynamic parameter allocation Function type malloc().
Function to abort and return to Operating System
Function abort ()
Aborts the current process, producing an abnormal program termination. The function raises the SIGABRT signal (as if raise(SIGABRT) was called). This, if uncaught, causes the program to terminate returning a platform-dependent unsuccessful termination error code to the host environment. soubor core.
Function atexit ()
The function pointed by func is automatically called without arguments when the program terminates normally.
If more than one atexit function has been specified by different calls to this function, they are all executed in reverse order as a stack (i.e. the last function specified is the first to be executed at exit).

atexit() function

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

void fnExit1 (void){
   printf ("\nExit function 1.");
}

void fnExit2 (void){
   printf ("\nExit function 2.");
}

int main (){

   atexit (fnExit1);
   atexit (fnExit2);
   printf ("Finish main function.");
   return 0;
}
```

exit() function

Terminates the process normally, performing the regular cleanup for terminating programs.
Normal program termination performs the following (in the same order): Objects associated with the current thread with thread storage duration are destroyed. Objects with static storage duration are destroyed and functions registered with atexit are called (if an unhandled exception is thrown terminate is called). All C streams (open with functions in ¡cstdio¿) are closed (and flushed, if buffered), and all files created with tmpfile are removed. Control is returned to the host environment.

system() function

Invokes the command processor to execute a command.

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

int main(void){
        printf ("Dir command is invoked from a program \n")
        system ("dir    . txt ");
        return 0;
}
```

string.h library

strcpy ()	Copies string from source to destination
strncpy ()	Copies the first num characters
strcat ()	Appends a copy of the string
strncat ()	Appends the first num characters
strcmp()	Compares two strings.
strncmp()	Compares up to num characters of two strings.
strchr ()	Returns a pointer to the first occurrence of character
strrchr ()	Returns a pointer to the last occurrence of character
strstr ()	Returns a pointer to the first occurrence
strlen ()	Returns the length of the string
strspn ()	Returns the length of the initial portion of string
strcspn ()	Scans string for the first occurrence of any of the characters

string.h library

strpbrk ()	Returns a pointer to the first occurrence in string
strerror ()	Interprets the value of errnum, generating a string
strtok ()	A sequence of calls to this function split str into tokens
memcpy()	Copies the values of num bytes from source
memmove()	Moves the values of num bytes from source
memcmp()	Compares the first num bytes of the block of memory
memchr()	Searches within the first num bytes of the block of memory
memset()	Sets the first num bytes of the block of memory

Memory Example

```
#include <stdio.h>
#include <string.h>
#define delka 1000

int main(void)
{
int a[delka],b[delka];
int i, *p_i;

memset((void *)a,0,(size_t)(delka* sizeof(int)));
printf("memset: a[%d] = %d \n", delka/2, a[delka/2]);
a[delka/2]=1;
p_i =(int *) memchr((void *) a, 1,(size_t)(delka * sizeof(int)));

printf("memchr: i=%u \n",p_i - a);

memcpy((void*)b,(void*)a,(size_t)(delka*sizeof(int)));
printf("memcpy: b[%d]=%d \n", delka/2 ,b[delka /2]);

for(i=1;  i < 9; i++)
  a[i]=i-1;
for(i=0 ; i < 9; i++)
printf("a[%d]=%d ",i,a[i]);

printf("\nmemmove: \n");
memmove((void*) &a[0],(void*) &a[1],(size_t)(9*sizeof(int)));

for(i=0; i < 9;i++)
printf("a[%d]=%d", i, a[i]);
return 0;
}
```

time.h library

clock()	Returns the number of clock ticks elapsed
time()	Get the current calendar time as a time_t object.
difftime ()	Calculates the difference in seconds
mktime()	Calendar time expressed in local time
asctime()	Readable version of calendar time
ctime()	Local time and date
gmtime()	Time as in GMY timezone
localtime ()	Corresponding local time
strftime ()	Structured form of time

CLK_TCK	Clock ticks per second
clock_t	usually long, for function clock(),
time_t	usually long, used for function clock(),
tm	for the time structure

time.h header

Structure of tm

int	tm_sec	seconds $(0, 59)$
int	tm_min	minutes $(0, 59)$
int	tm_hour	hours $(0, 23)$
int	tm_mday	days in month $(1, 31)$
int	tm_mon	months from January $(0, 11)$
int	tm_year	years since 1970 $(0,)$
int	tm_wdaz	days from Monday $(0, 6)$
int	tm_ydaz	days from 1. January $(0, 365)$
int	tm_isdst	summer time

time.h example

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

using namespace std;
int main(int argc, char *argv[]) {

        clock_t  zac, konec;

        zac =clock();
        konec = clock();
        printf("program trval :  %f [sec] \n",(konec-zac)/(double) CLOCKS_PER_SEC);

        return 0;
}
```

time.h example

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

#define shift 20000

int main(void){

        struct tm *p_pak;
        time_t ted;

        char*day[]={"Mo","Tu","We","Th","Fr","Sa","Su"};
        time(&ted);
        p_pak=localtime( &ted);
        p_pak->tm_min += shift;

        mktime(p_pak);
        printf("In %d it will be %s. \n",shift,day[p_pak->tm_wday]);

        return 0;
}
```