Procedūrinio programavimo pagrindai

Bibliotekos

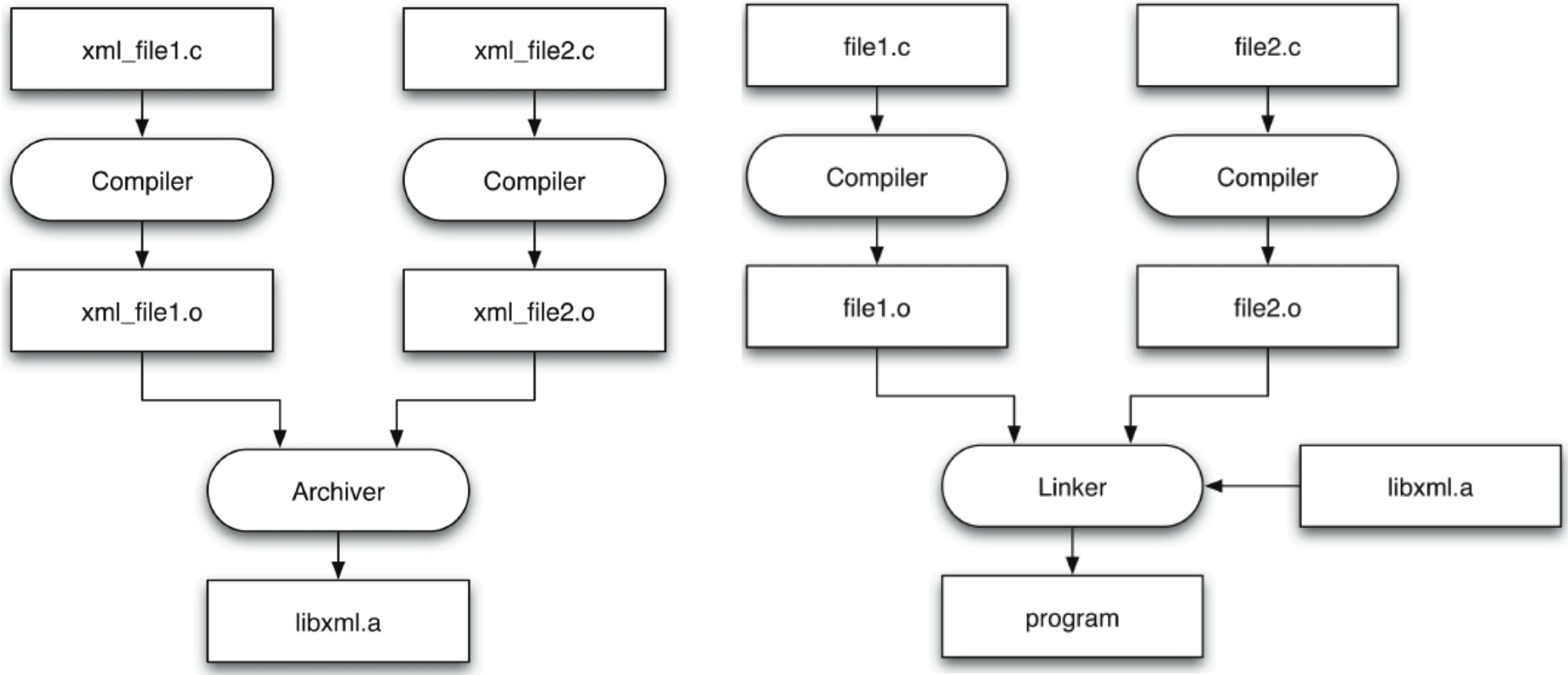
lekt. Irmantas Radavičius
irmantas.radavicius@mif.vu.lt
Informatikos institutas, MIF, VU

Turinys

Bibliotekos sąvoka

Biblioteka – tai resursų rinkinys programinei įrangai kurti.

Jos viduje – konstantos, kintamieji, duomenų tipai, funkcijos, etc.



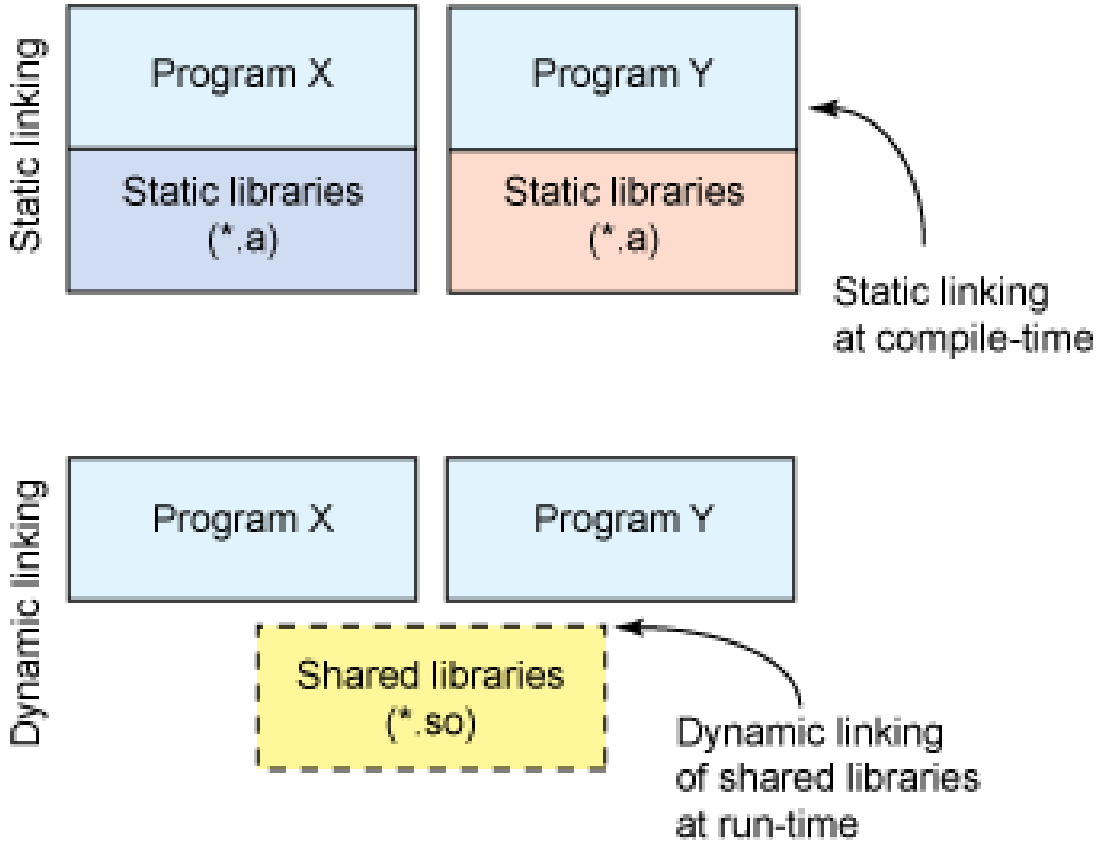
Statinės ir dinaminės bibliotekos

Statinė biblioteka
(archyvas)

- .a
- .lib

Dinaminė biblioteka
(bendrasis objektas)

- .so
- .dll



Statinė biblioteka

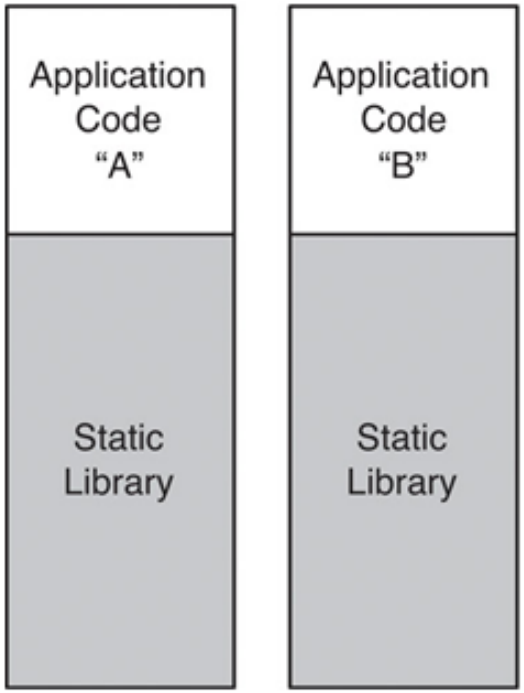
Biblioteka naudojama kompiliavimo proceso metu.

Kiekvieno vykdomojo failo viduje – bibliotekos fragmento kopija.

Naudojami tik tie object failai, kurie yra reikalingi!

Savybės:

- – auga vykdomojo failo dydis
- – po pakeitimų reikia jį perkompiliuoti
- ✓ visas reikalingas kodas – programos viduje
- ✓ santykinai nepriklauso nuo OS



Dinaminė biblioteka

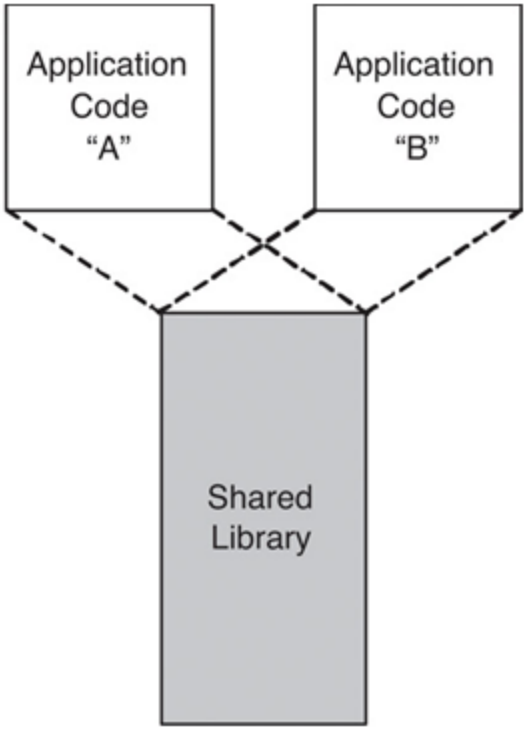
Biblioteka naudojama vykdymo proceso metu.

Kelios programos naudoja tą patį kodą, esantį kitame faile.

Procesas priklauso nuo OS realizacijos.

Savybės:

- ✓ mažesnis vykdomojo failo dydis
- ✓ po pakeitimų pakanka atnaujinti biblioteką
- – neveiks, jei bibliotekos nėra arba ji nedera
- – efektyvumas priklauso nuo OS



Bibliotekų kūrimas (GCC et al. 4 Win)

Statinė biblioteka

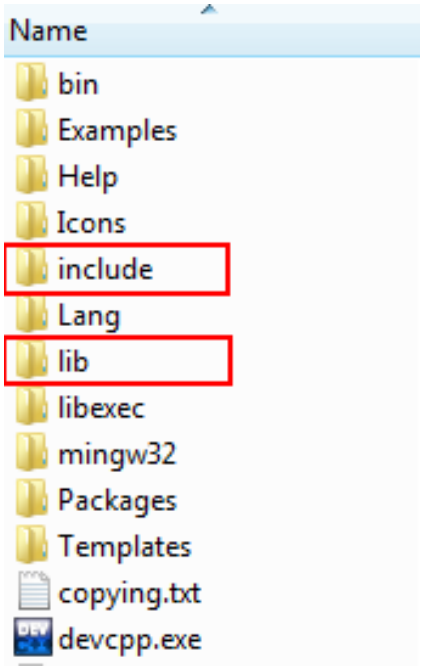
```
gcc –c file1.c –o file1.o
gcc –c file2.c –o file2.o
ar rc libsomename.a file1.o file2.o
gcc main.c –L. –lsomename –o main.exe
main.exe
```

Dinaminė biblioteka

```
gcc –c file1.c –o file1.o
gcc –c file2.c –o file2.o
gcc –shared file1.o file2.o –o libsomename.dll
gcc main.c –L. –lsomename –o main.exe
main.exe
```

C standartinė biblioteka

Vartotojo sukurtos (angl. custom) bibliotekos



Standartinė C biblioteka

- pradžioje – 15 antraštės failų (C89)
- C99: <stdbool.h> <complex.h> ...
- C11: <uchar.h> <threads.h> …
- dabar – 29 antraštės failai (C11)

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

Stdio.h – darbas su srautais

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

- failų manipuliavimo funkcijos (fopen, fclose, ...)
- pozicionavimo faile funkcijos (ftell, fseek, ...)
- buferių valdymo funkcijos (setvbuf, fflush, ...)
- formatuoto I/O funkcijos (fprintf, fscanf, ...)
- tekstinio I/O funkcijos (fgetc, fputc, fgets, fputs, ...)
- atminties blokų I/O funkcijos (fread, fwrite)

Stdlib.h – įvairios tarnybinės funkcijos

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

- dinaminės atminties skirstymas (malloc, free, ...)
- konvertavimas iš teksto į skaičius (atoi, atol, atof, ...)
- atsitiktinių skaičių generavimas (srand, rand)
- sisteminės funkcijos (system, exit, atexit, ...)
- duomenų apdorojimas (bsearch, qsort)
- kitos (abs, div, ...)

Stdarg.h – kintamas parametrų sąrašas

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

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

/* print all non-negative args one at a time;
   all args are assumed to be of int type */
void printargs(int arg1, ...)
{
  va_list ap;
  int i;

  va_start(ap, arg1);
  for (i = arg1; i >= 0; i = va_arg(ap, int))
    printf("%d ", i);
  va_end(ap);
  putchar('\n');
}
```

Assert.h – klaidų prevencija

<assert.h>	**<float.h>**	**<math.h>**	**<stdarg.h>**	**<stdlib.h>**
<ctype.h>	**<limits.h>**	**<setjmp.h>**	**<stddef.h>**	**<string.h>**
<errno.h>	**<locale.h>**	**<signal.h>**	**<stdio.h>**	**<time.h>**

```c
/* assert example */
#include <stdio.h>
#include <assert.h>

void print_number(int* myInt) {
  assert (myInt!=NULL);
  printf ("%d\n",*myInt);
}
```

```c
int main ()
{
  int a=10;
  int * b = NULL;
  int * c = NULL;

  b=&a;

  print_number (b);
  print_number (c);

  return 0;
}
```

Jei išraiška lygi 0,

programa nutraukiama.

Limits.h – sveikųjų skaičių rėžiai

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

CHAR_BIT	8	bits in a char
CHAR_MAX	UCHAR_MAX *or* SCHAR_MAX	maximum value of char
CHAR_MIN	0 *or* SCHAR_MIN	minimum value of char
INT_MAX	+32767	maximum value of int
INT_MIN	-32767	minimum value of int
LONG_MAX	+2147483647	maximum value of long
LONG_MIN	-2147483647	minimum value of long
SCHAR_MAX	+127	maximum value of signed char
SCHAR_MIN	-127	minimum value of signed char
SHRT_MAX	+32767	maximum value of short
SHRT_MIN	-32767	minimum value of short
UCHAR_MAX	255	maximum value of unsigned char
UINT_MAX	65535	maximum value of unsigned int
ULONG_MAX	4294967295	maximum value of unsigned long
USHRT_MAX	65535	maximum value of unsigned short

Float.h – realiųjų skaičių rėžiai

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

DBL_DIG	10	decimal digits of precision
DBL_EPSILON	1E-9	smallest number x such that $1.0 + x \neq 1.0$
DBL_MANT_DIG		number of base FLT_RADIX digits in mantissa
DBL_MAX	1E+37	maximum double floating-point number
DBL_MAX_EXP		maximum n such that FLT_RADIX$^n-1$ is representable
DBL_MIN	1E-37	minimum normalized double floating-point number
DBL_MIN_EXP		minimum n such that 10^n is a normalized number
FLT_RADIX	2	radix of exponent representation, e.g., 2, 16
FLT_ROUNDS		floating-point rounding mode for addition
FLT_DIG	6	decimal digits of precision
FLT_EPSILON	1E-5	smallest number x such that $1.0 + x \neq 1.0$
FLT_MANT_DIG		number of base FLT_RADIX digits in mantissa
FLT_MAX	1E+37	maximum floating-point number
FLT_MAX_EXP		maximum n such that FLT_RADIX$^n-1$ is representable
FLT_MIN	1E-37	minimum normalized floating-point number
FLT_MIN_EXP		minimum n such that 10^n is a normalized number

Ctype.h – simbolių apdorojimas

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

`isalnum(c)`	`isalpha(c)` or `isdigit(c)` is true
`isalpha(c)`	`isupper(c)` or `islower(c)` is true
`iscntrl(c)`	control character
`isdigit(c)`	decimal digit
`isgraph(c)`	printing character except space
`islower(c)`	lower-case letter
`isprint(c)`	printing character including space
`ispunct(c)`	printing character except space or letter or digit
`isspace(c)`	space, formfeed, newline, carriage return, tab, vertical tab
`isupper(c)`	upper-case letter
`isxdigit(c)`	hexadecimal digit

`int tolower(int c)`	convert `c` to lower case
`int toupper(int c)`	convert `c` to upper case

String.h (1) – eilučių apdorojimas

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

`char *strcpy(s,ct)`	copy string `ct` to string `s`, including `'\0'`; return `s`.
`char *strncpy(s,ct,n)`	copy at most `n` characters of string `ct` to `s`; return `s`. Pad with `'\0'`s if `t` has fewer than `n` characters.
`char *strcat(s,ct)`	concatenate string `ct` to end of string `s`; return `s`.
`char *strncat(s,ct,n)`	concatenate at most `n` characters of string `ct` to string `s`, terminate `s` with `'\0'`; return `s`.
`int strcmp(cs,ct)`	compare string `cs` to string `ct`; return <0 if `cs<ct`, 0 if `cs==ct`, or >0 if `cs>ct`.
`int strncmp(cs,ct,n)`	compare at most `n` characters of string `cs` to string `ct`; return <0 if `cs<ct`, 0 if `cs==ct`, or >0 if `cs>ct`.
`char *strchr(cs,c)`	return pointer to first occurrence of `c` in `cs` or `NULL` if not present.
`char *strrchr(cs,c)`	return pointer to last occurrence of `c` in `cs` or `NULL` if not present.

String.h (2) – eilučių apdorojimas

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

`size_t strspn(cs,ct)`	return length of prefix of `cs` consisting of characters in `ct`.
`size_t strcspn(cs,ct)`	return length of prefix of `cs` consisting of characters *not* in `ct`.
`char *strpbrk(cs,ct)`	return pointer to first occurrence in string `cs` of any character of string `ct`, or `NULL` if none are present.
`char *strstr(cs,ct)`	return pointer to first occurrence of string `ct` in `cs`, or `NULL` if not present.
`size_t strlen(cs)`	return length of `cs`.
`char *strerror(n)`	return pointer to implementation-defined string corresponding to error `n`.
`char *strtok(s,ct)`	`strtok` searches `s` for tokens delimited by characters from `ct`; see below.

String.h (3) – eilučių apdorojimas

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

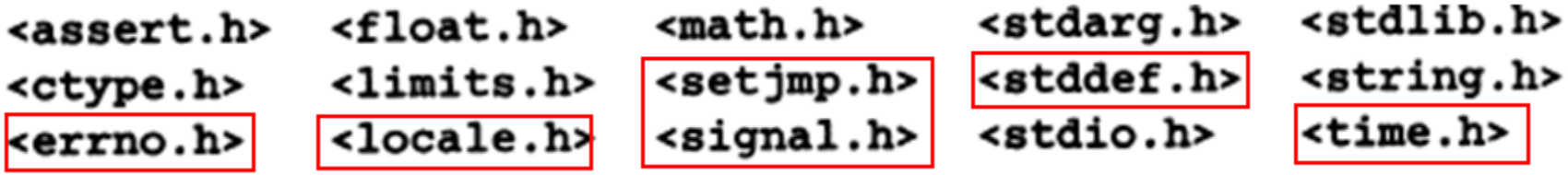
`void *memcpy(s,ct,n)`	copy `n` characters from `ct` to `s`, and return `s`.
`void *memmove(s,ct,n)`	same as `memcpy` except that it works even if the objects overlap.
`int memcmp(cs,ct,n)`	compare the first `n` characters of `cs` with `ct`; return as with `strcmp`.
`void *memchr(cs,c,n)`	return pointer to first occurrence of character `c` in `cs`, or `NULL` if not present among the first `n` characters.
`void *memset(s,c,n)`	place character `c` into first `n` characters of `s`, return `s`.

Math.h – matematinės funkcijos

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

- trigonometrinės funkcijos (sin, cos, tan, ...)
- kėlimas laipsniu (sqrt, exp, pow, ...)
- apvalinimas (floor, ceil, ...)
- kitos funkcijos (log, fabs, ...)

Įvairūs



<errno.h>	-	klaidų apdorojimas
<locale.h>	-	lokalizacijos priemonės
<setjmp.h>	-	nelokalių šuolių įgyvendinimas
<signal.h>	-	signalų apdorojimas
<stddef.h>	-	kai kurie apibrėžimai (pvz. NULL)
<time.h>	-	datos ir laiko funkcijos

Pavyzdys: <errno.h>

```c
/* strerror example : error list */
#include <stdio.h>
#include <string.h>
#include <errno.h>

int main ()
{
  FILE * pFile;
  pFile = fopen ("unexist.ent","r");
  if (pFile == NULL)
    printf ("Error opening file unexist.ent: %s\n",strerror(errno));
  return 0;
}
```

Pavyzdys: <setjmp.h>

```c
/* longjmp example */
#include <stdio.h>
#include <stdlib.h>
#include <setjmp.h>

main()
{
  jmp_buf env;
  int val;

  val=setjmp(env);

  printf ("val is %d\n",val);

  if (!val) longjmp(env, 1);

  return 0;
}
```

```
val is 0
val is 1
```

Pavyzdys: <signal.h>

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

char tmpfilename [L_tmpnam];

void terminate (int param)
{
  printf ("Terminating program...\n");
  remove (tmpfilename);
  exit(1);
}
```

```c
int main ()
{
  void (*prev_fn)(int);

  prev_fn = signal (SIGTERM,terminate);
  if (prev_fn==SIG_IGN) signal (SIGTERM,SIG_IGN);

  tmpnam (tmpfilename);

  /* code here */

  return 0;
}
```

Pavyzdys: <locale.h> ir <time.h>

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

int main ()
{
  time_t rawtime;
  struct tm * timeinfo;
  char buffer [80];

  struct lconv * lc;

  time ( &rawtime );
  timeinfo = localtime ( &rawtime );

  int twice=0;

  do {
    printf ("Locale is: %s\n", setlocale(LC_ALL,NULL) );

    strftime (buffer,80,"%c",timeinfo);
    printf ("Date is: %s\n",buffer);

    lc = localeconv ();
    printf ("Currency symbol is: %s\n-\n",lc->currency_symbol);

    setlocale (LC_ALL,"");
  } while (!twice++);

  return 0;
}
```

```
Locale is: C
Date is: 01/15/07 13:33:47
Currecy symbol is:
-
Locale is: English_United States.1252
Date is: 1/15/07 1:33:47 PM
Currency symbol is: $
-
```

<bye.h>