FEJLETT PROGRAMOZÁSI NYELVEK, 2009
1. GYAKORLAT - Linux alatti C programozás

Cél:

- A gcc (GNU C Compiler) néhány kapcsolója (-c,-o,-I,-L,-l)
- Statikus csatolású függvénykönyvtár készítése és használata
- A make segédprogram használata, Makefile készítése

1. Kapcsolók

A GNU C fordító a fejlécállományokat a következő szabványos katalógusokban keresi:

```
/usr/include
/usr/local/include
```

Ha nincs írási jogosultságunk a fenti katalógusokhoz, illetve ha máshol helyezzük el fejállományainkat, akkor a fordítónak -I kapcsoló után megadható egy katalóguslista, amelyben a fejállományokat keresi. Ebben az esetben először itt keres, majd ha itt nem találja, akkor a szabványos katalógusokban. Például:

```
gcc foo.c -I /home/students/janos/cprog/include
```

A GNU C fordító automatikusan meghívja az ld szerkesztő (linker) programot a végrehajtható állomány szerkesztéséhez, kivételt képez a -c kapcsolóval hívott eset, ilyenkor csak fordítás történik. Az ld program a függvénykönyvtárakat a következő szabványos katalógusokban keresi:

```
/lib
/usr/lib
/usr/local/lib
```

Ha a függvénykönyvtár nem a szabványos katalógusokban található, a -L kapcsolóval megadható egy katalóguslista. Ebben az esetben a megadott katalóguslistában keres először. A **-l** kapcsolóval a függvénykönyvtár neve adható meg. Például, ha van egy `libsajat.a` statikus csatolású függvénykönyvtárunk a `/home/students/janos/cprog/lib` katalógusban és egy `main.c` állományunk, amelyben használjuk a függvénykönyvtárbeli függvényeket, akkor a következő paranccsal kapjuk meg a végrehajtható állományt:

```
gcc main.c -L /home/students/janos/cprog/lib -lsajat
```

Feladat

Írassa ki a szinusz függvény értékeit a következő pontokban: $0, \pi/4, \pi/3, \pi/2$

A matematikai függvénykönyvtár csatolását nem végzi automatikusan a szerkesztő!!! `(-lm)`

2. Statikus csatolású függvénykönyvtár készítése

Készítsen egy *verem* adatstruktúra modult. Adott a fejállomány és egy példaprogram a használatra. A feladat a fejállományban deklarált struktúra és függvények megvalósítása (implementálása).

stack.h

```
#ifndef _STACK_H
#define _STACK_H

typedef struct stack * STACKPTR;
typedef double ELEMTYPE;

STACKPTR create( int );
void destroy( STACKPTR );
void push( STACKPTR, ELEMTYPE);
ELEMTYPE pop( STACKPTR );
int isempty( STACKPTR );

#endif
```

main.c

```
#include "stack.h"
#include <stdio.h>

main(){
     int i;
     STACKPTR s = create( 10 );
     for( i=0; i<10; i++ )
          push( s, i );
     while( ! isempty( s ) )
          printf("%lf\t", pop( s ) );
     destroy( s );
     return 0;
}
```

1. lépés

Készítse el a `stack.c` forrásállományt. Fordítsa le és hajtsa végre a programot!

```
gcc -c stack.c
gcc -c main.c
gcc  main.o stack.o -o verem
./verem
```

vagy

```
gcc  main.c stack.c
./a.out
```

2. lépés

Készítsen egy *statikus csatolású függvénykönyvtárat* a stack modulból.

```
ar -r libstack.a stack.o
ranlib libstack.a
```

3. lépés

Fordítsa újra a `main.c` állományt úgy, hogy az előző lépésben elkészített függvénykönyvtárat használja.

```
gcc main.c -lstack -L .
./a.out
```

3. make és Makefile

- Olvassa el a Makefile készítésének lépéseit.
- Vezérelje az 1. lépésben készített program (stack.h, stack.c, main.c) fordítását *Makefile* segítségével.