Sistemi Operativi: Modulo Laboratorio a.a. 14/15 Corso-A

Creazione di librerie statiche e dinamiche in programmi C

Tipi di librerie

- Una libreria è un file archivio contenente codice compilato che verrà agganciato al programma nella fase di linking
- Le librerie sono utilizzate prevalentemente per:
  - Ridurre i tempi di compilazione
  - Diminuire la size dei programmi
  - Diminuire l'occupazione di memoria in programmi che utilizzano la stessa libreria (per librerie dinamiche)
- Ci sono 2 tipi di librerie:
  - Statiche (estensione “.a”)
  - Condivise o Dinamiche (estensione “.so”)

Librerie statiche (.a)

- La libreria è agganciata al programma allatto della compilazione per formare un eseguibile monolitico.
  - Creazione dei file oggetto:

    gcc -std=c99 -Wall myfunc.c -c -o myfunc1.o

    gcc -std=c99 -Wall myfunc2.c -c -o myfunc2.o

    gcc -std=c99 -Wall errfunc.c -c -o errfunc.o
  - Creazione della libreria statica:

    ar rvs libmyf.a myfunc1.o myfunc2.o errfunc.o
  - Linking e creazione dell'eseguibile:

    gcc main.c -o myprog -L . -lmyf

    oppure

    gcc main.c -o myprog $pwd/libmyf.a

Librerie Condivise (.so)

- La libreria è agganciata al programma in 2 fasi. A tempo di compilazione il linker verifica che tutti i simboli siano definiti; nella fase di carimento il loader carica le librerie agganciate al programma (quelle necessarie).
  - Creazione dei file oggetto:

    gcc -std=c99 -Wall myfunc.c -c -fPIC -o myfunc1.o

    ...
  - Creazione della libreria dinamica:

    gcc -shared -o libmyf.so myfunc1.o ….
  - Linking e creazione dell'eseguibile:

    gcc main.c -o myprog -L . -lmyf

-fPIC

Generate position-independent code (PIC) suitable for use in a shared library, if supported for the target machine.

Position Independent Code means that the generated machine code is not dependent on being located at a specific address in order to work.

In computing, position-independent code (PIC) or position-independent executable (PIE) is a body of machine code that, being placed somewhere in the primary memory, executes properly regardless of its absolute address. PIC is commonly used for shared libraries, so that the same library code can be loaded in a location in each program address space where it will not overlap any other uses of memory (for example, other shared libraries).

-fPIC – example

PIC: funziona con il codice caricato a indirizzo 100 o 1000

```
100: COMPARE REG1, REG2
101: JUMP_IF_EQUAL CURRENT+10
...
111: NOP
```

NON-PIC: funziona solo con il codice caricato a indirizzo 100

```
100: COMPARE REG1, REG2
101: JUMP_IF_EQUAL 111
...
111: NOP
```

-fPIC

If your code is compiled with **-fPIC**, it's suitable for inclusion in a library

- The library must be able to be relocated from its preferred location in memory to another address
  - There could be another already loaded library at the address your library prefers

Loading di librerie condivise

- Lanciamo il programma $> ./myprog

  ./myprog: error while loading shared libraries: libmyf.so:
  cannot open shared object file: No such file or directory

- $> ldd ./myprog

  linux-vdso.so.1 => (0x00007fff597fe000)
  libmyf.so => not found
  libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f4cef03b000)
  /lib64/ld-linux-x86-64.so.2 (0x00007f4cef42a000)

- Per dire al loader dove trovare le librerie si può usare

  export LD_LIBRARY_PATH="./:${LD_LIBRARY_PATH}"

  di default il dynamic linker cerca nei path specificati in /etc/ld.so.conf* ed in /lib /usr/lib (**vedere man ldconfig**)

- Si può anche usare l'opzione del loader **-rpath**

  gcc main.c -o main -Wl,-rpath,./ -L. -lmyf

Qui in generale metto il path dove saranno le mie librerire