

* Session 3 *

* بمن الممكن في أول نمادن session قد حملت المكتبة static lib. static lib يمكّن إدخاله static lib. static lib. shared (dynamic) lib.

Static lib:

* الباقي static lib يمكنه إنتاج code كل الـ program يستخدمها لأنها مبنية على كل المكتبات static lib التي تتعارض مع كل المكتبات static lib التي تم إدخالها في المكتبات static lib.

* فمثلاً يتعارض static lib stdio.h مع static lib stdio.o وظائف static lib stdio.o التي تم إدخالها في المكتبات static lib stdio.h.

* طلب أنا استدفعت أي وقت Compilation

Memory دیکھاں

* طب ابي مخواص الـ Staticlib
* ان مع كل printf فيه file.c كود ان .stdio.h والـ printf ينزل في الـ Output معاً كده انت رقم 1 في redundancy في الـ code معاً له لو حملت 5 files.c موجودين في المجلد مع كل file.c كود stdio.h و كل printf في المجلد first كل files الى مستخدمها من الاول فليس يعني الكلاس ده مع kernel في الـ hardware مفرق جداً احنا هنا هنا يعني فيه ميزة واحدة ان انت ضاشر ان الـ code معاً معهون زي ما هو فالـ hardware safe لان لو فالـ library error مثل موجودة انت في مطابلك

* المروض معه لاحتياجاته
embedded desktop أو ما يسمى بالـ الجهاز
وذلك يستخدم embedded linux ولقد
يستخدمونه في الكمبيوتر المكتبي أو الكمبيوتر المحمول

Wohin ist der Weg? (richtig)

How to make static linking? *

also Compile \rightarrow LibC \rightarrow *

dynamically \leftarrow Compile \rightarrow by default

أنا هنا أشرح \rightarrow example \rightarrow *

one session \rightarrow session \rightarrow \rightarrow

myadd.c \rightarrow mylib \rightarrow directory

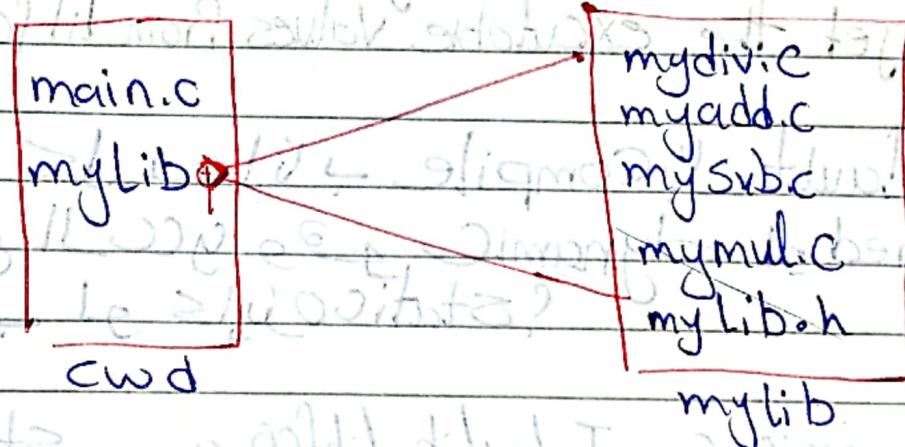
mymul.c \rightarrow mydiv.c \rightarrow mySub.c

روابط \rightarrow واجهة \rightarrow Functions

prototypes \rightarrow وارتفع \rightarrow ونهاية

mylib.h \rightarrow mySub.c \rightarrow Functions

functions \rightarrow main.c \rightarrow main.c



نحو \rightarrow files \rightarrow build \rightarrow نجرب \rightarrow *

gcc main.c ./mylib*.* -I./mylib

include paths for linker \rightarrow

mylib.h مكان \rightarrow

طبع المفهوم ، لو انا ابي
file 100 طبع المفهوم ، لو انا ابي
files في المفهوم بالطريقة
Compile files into archive
بمعنى ال files.o بنهاية

ar -r libfoo.a *.o

↳ add all the files.o to an archive named
libfoo.a

↳ update libfoo.a
للحفاظ على جميع الملفات

ar -t libfoo.a

↳ list the table of Content of libfoo.a

gcc main.c -I:/mylib libfoo.a

↳ get the executable values from libfoo.a

default كل الملفات بـ

Shared او Dynamic بـ gcc

طبع المفهوم ، لو انا ابي

gcc main.c -I:/mylib libfoo.a -static

↳ Compile statically

لو حبيت تعمل ls -lh انت
 الحجم بتاع main.out اوال a.out
 كبر او اوى او اوى ودى تقريرا
 المشكلات الاقعه جمله حوار انت
 محتاج تتنزيل file! يستخدم
 static libraries

ولو كملت او a.out على ال obdmp
 وحبيت شوري على حاجة دوا ال libc
 حق لو انت مستخدمتو ايش فنادق
 كاردي ما لو ما حملت obdmp على
 Static Libraries a.out
 وحيث سearched sysCall متى دخلت
 هنا فيها

2019-10-20

2019-10-20

dynamic (shared) libraries

* انت لو كنت في حاجة Common Libraries و في أكثر من program ي المستخدمة زع المكان studio ليس اروح اعطيك مع كل ممكنة ما هو program وكل ال programs اللي يستخدمها يعني عارفة مكانها طبعاً بعمل الدمار به ازاي ؟

* يروح اعمل Shared library بعدين طرق اتنالطل
بروح اعمل Files اللى Compilt اكملوا
PIC ← gcc بس فدى ال Compile
position independant option
يعنى ال code اللى فيطلع he لا يكتفى
على المكان يتنادى في memory, حيثان اع
Program كابن يستخدم

gcc -c *.c -fPIC

* الـP/O بناء الـCommand فيكون في الـmemory في المكان المخصص له وـPiles هي التنفيذ

* بعد كده أناحتاج اجمع الـ Object Files

رجـ PIC رـي في Library اـيـ استـعـدـ مـعـ

بسـ خـلـيـ بـالـلـكـ أـنـ أـسـعـدـ لـذـمـ

يـقـيـ اـولـ اـصـطـبـ وـ خـلـيـ بـالـلـكـ بـرـقـ وـ أـنـ مـنـ

shared object

gcc -fPIC -shared *.o -o libfoo.so

* مـعـ دـرـرـشـ archive رـي مـتـزـيـ الـ libfoo.so

اـكـلـ مـنـهـاـ اوـ اـيـ حـاجـتـ لـذـمـ اـيـلـ build

الـ اـولـ تـأـيـ ،ـ قـلـ مـاـ اـكـمـلـ فـيـ حـاجـتـ مـعـهـ

dynamic Loader

dynamic loader

* اـنـ زـمـانـ اـمـاـ كـيـتـعـدـ بـتـقـاسـلـ معـ الـ PIC

لـوـ فـاكـرـ اـنـ مـنـضـمـ الـ build

fileـsـ كـيـتـعـدـ بـتـرـقـ تـكـمـلـ الـ PIC

كـلـهـاـ بـسـ لـذـهـاـ اـنـ بـتـكـلمـ عـاـلـ

Compilation process

Compile time linking

linux | runtime link

والـ windows شـيـاـلـنـ كـرـهـ ،ـ طـبـ كـرـهـ اـنـ اـنـ

dynamic loader

عوكلوس dynamic Loader

لینکر (Linker) یا files ایجاد کردن

تحتاجها في runtime كن طريق انه

بروج يدور في ملائكة system directory (ف)

LD_LIBRARY_PATH as 1 Variable is this

الآن أنت تعلمها وتحولها إلى أرثى دائم في ذهنك

tail slightly raised to 10-15% body length

* فاست کہ اول حاجت محتاج تھیف

الرجاء إدخال libfoo.50 في المجلدة

الدول كعنوان dynamic loader يأخذ عنوان

⇒ LD_LIBRARY_PATH = ./mylib

⇒ `export LD_LIBRARY_PATH=.:mylib`

* **نکاتی خلاصه‌برداری اما حفظیات** directory

الى System directories جوا mylib

main ॥ Compiled by libfoo.so

gcc -fno-ld.l./mylib main.o /l.xm/

ell zentralisierter Zirkulation der Zahlabilität
H = Library number = library code (catalogue code)

Note 8-

inside all dynamic loader ||
Kernel ||

Variadic Function

Variable-أدخال Function is also Variadic Function \Rightarrow *
one printf \Rightarrow زائد number of arguments.

Syntax: int add(int n, ...)

* الاول: لأنها تروح تتحمل pointer يساور على va-list رعاي من نوع arguments

→ va_list Varptr;

* يرجى ملاحظة أن `ptr` هو متغير يشير إلى `initialize`، وأن `ptr` يشير إلى `Parameters`.

~~va_start~~ (varptr, n);

↳ Macros

* ما تکون تجیب va-arg چیزی arguments

ورى كل ماتند هو، فتبيّنك الـ argument الى اللي في
وايّقت على اللي بعده ولازم تدروا الفكرة

→ Va-arg (Varptr; int)

Il s'agit d'un Macro.

ptr JI Free õ üüs Va - end pisiwüüd Galüü lo! *

→ Va_end (parptr);
→ Macros

Example 8-

```

#include <stdio.h>
#include <stdarg.h>
int add (int arg_num, ...){}
    int sum = 0;
    va_list ptr;
    va_start (ptr, arg_num);
    for (int i = 0; i < n; i++)
        sum = va_arg (ptr, int);
    va_end (ptr);
    return sum;

```

34

```

int main(){
    add (5, 1, 2, 3, 4, 5); → 15
    add (6, 1, 2, 3, 4, 5, 6); → 21
    add (4, 1, 2, 3, 4); → 10
    return 1;
}

```

34

Note :-

is legal behavior || va_end || *
 argument || return || elastic architecture ||
 CPU reg. || Stack || is Function || call

Chapter 5

* الدليل Flags في File desc. Flags يُعرف بـ flags
روابط الـ Properties

* كل data structure في Inode موجود في كل directory أو file.
الاحتاجات في قدرها في الملفات المترابطة.

لوات

* افتح مكالماً صحفة في المراجع إلى في gie
الـ file descriptor table.

File descriptor table:-

* كل Process يكون لها File descriptor table يحتوي على指针指向 readwrite و readonly و readonly flags و open file table.

* يمكن اكتشاف File descriptor table في process table.
الـ file descriptor table يخزن معلومات عن نفس الملف.

* يمكن اكتشاف File descriptor table في processes table.
الـ child process له واحدة open file table.
والـ parent process له فتحة fork.

Open File Table:

هذا هو table يحتوي على كل files الذي موجودة حالياً أو مش مفتوحة، ال files الذي موجودة حالياً يعني في system بتاتاً

يكون فيه ال offset file للfiles الذي pointer وهو status flags والstatus flags هو مفتاح file لـ inode

status Flags vs Fd Flags

ال status flags هي ملزمه ال file flags هي الأولى ما تتم لحد ما انت حنفت من sys زى APPEND التي بتخلي كل ما تجي تفتح ال file يفتح في ال mode offset هو كان فيه

اما ال Fd Flags التي خاصة بالفتحة رعايس ال file O_RDONLY ، زى O_RDONLY ، يفتح المفتوحة file القراءة فقط ، ينساين ممكن تفتح المرة الجاية بـ O_WRONLY ، مثل و كذا

I-node Tables

* دو table فيه يبني inode بناء كل الملفات
الى في system ل OGK رقا الـ فيها نوع الملف و
كل الملفات metadata بناء كل الملفات

5.5 duplicating file descriptors

* ملخص I/O redirection هنا ، لو انت تأهل الحوار هنا ،
الـ file error file يفتح بـ I/O redirection ،
فـ file Process موجود في الـ I/O مند للـ file
مرة تـ استـ يـ بـ duplicates ، dub2 - dub1
مرة تـ استـ يـ بـ overwite وـ cd بـ file

6 read ch.6 & ch.41 From 41.1 to 41.4.3d