Centro de
Investigación en
Computación

Tool chain

Herramientas de desarrollo

Precompilador
Compilador
Ligado

Course

Operating System (with focus on Security)

Instructor

Acosta Bermejo Raúl

Lecture notes

Instituto
Politécnico
Nacional

Table of contents (outline)

Tabla de contenido

Introduction

Introducción

Toolchain

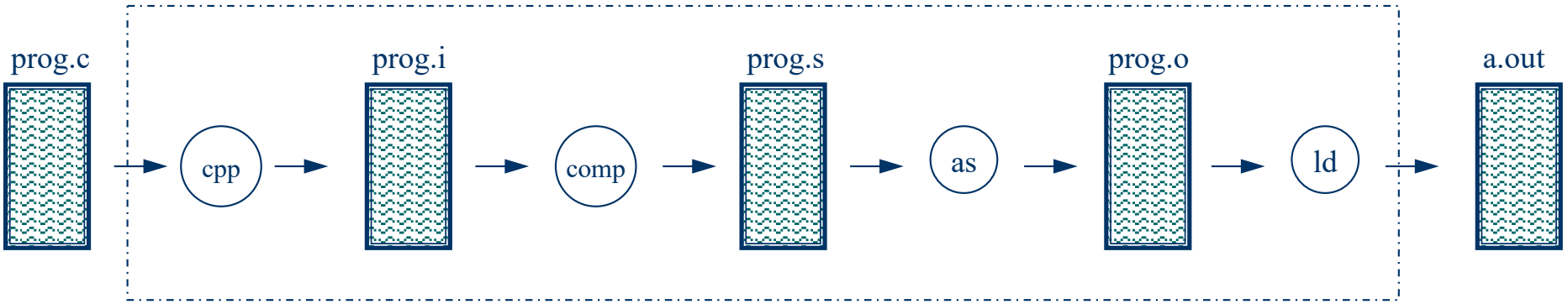
- It is the **set of programming tools** that is used to perform a complex software development task or to create a software product, which is typically another computer program or a set of related programs.
- In general, the tools forming a toolchain are executed **consecutively** so the output or resulting environment state of each tool becomes the input or starting environment for the next one, but the term is also used when referring to a set of related tools that are not necessarily executed consecutively.
- A simple software development toolchain may consist of:
  - A compiler and linker (which transform the source code into an executable program),
  - Libraries (which provide interfaces to the operating system), and
  - A debugger (which is used to test and debug created programs).

Introduction

Secuencia de pasos

Toolchain

cc Front-end para varios compiladores: C, C++, Objective-C, etc.



`cpp`	Preprocesamiento (Macros).
`comp`	Compilación y optimización.
`as`	Generación de código objeto.
`ld`	Enlace de bibliotecas.

```
$ cc p1.c
$ cc -o p1 p1.c
```

Archivo ejecutable **a.out**

Renombrado del ejecutable a **p1**

Introduction

Secuencia de pasos

Toolchain

Opciones de compilación

- Preprocesamiento (Macros)
  - prog.c -> prog.i.
  - $ gcc -E hola.c
- comp Compilación y optimización.
  - prog.i -> prog.s.
  - $ gcc -S prog.c
- as Generación de código objeto del ensamblador.
  - prog.s -> prog.o.
  - $ gcc -c

Option –v
Information about
- Compiler
- Path
- Version

Introduction

Secuencia de pasos

Toolchain
Other options

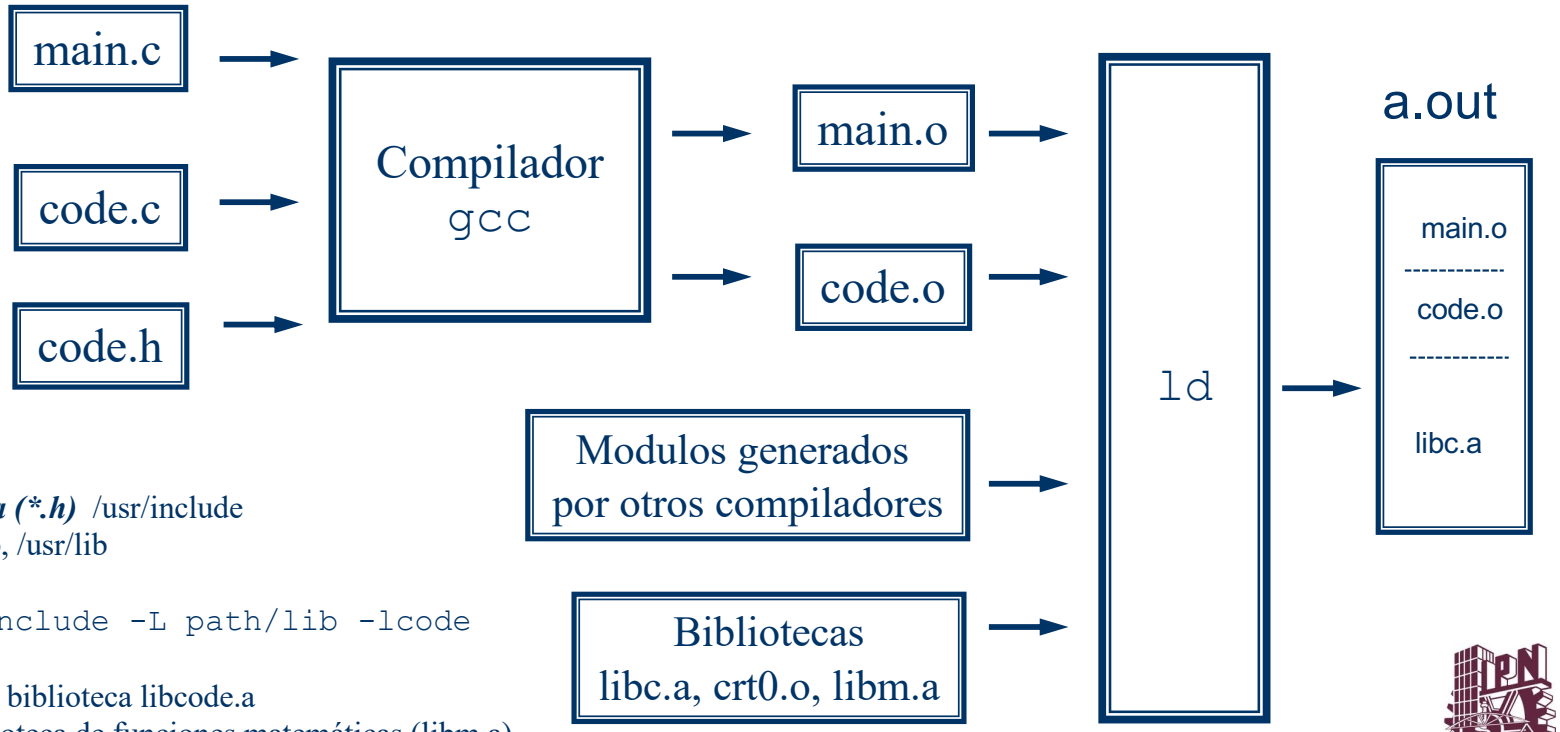
- Optimization
  Reduce code size and execution time: -O0, -O1, etc.

Compiling

Secuencia de pasos

Toolchain

Ligado de las bibliotecas



Archivos de cabecera (*.h) /usr/include
Bibliotecas (*.a) /lib, /usr/lib

```
$ cc -I path/include -L path/lib -lcode
```

`-lcode` Busca la biblioteca libcode.a
`-lm` Busca la biblioteca de funciones matemáticas (libm.a)

Introduction

Toolchain

Toolchain

Cross compiling

- A **cross compiler** is a compiler capable of creating executable code for a platform other than the one on which the compiler is running.
- For example, a compiler that runs on a Windows 7 PC but generates code that runs on Android smartphone is a cross compiler.
- The fundamental use of a cross compiler is to separate the build environment from target environment. This is useful in a number of situations:
  - Embedded computers
  - Compiling on a server farm
  - Bootstrapping
- Possible platforms are:
  - IA-32
  - x86_64
  - ARM
  - 32-bit Power PC
  - 64-bit Power PC

Introduction

Toolchain

Toolchain

Tipos de librerias y de ligado (link)

Existen 2 tipos de ligado:

- Estático
  - A static library or statically-linked library is a set of routines, external functions and variables which are resolved in a caller at compile-time and copied into a target application by a compiler, linker, or binder, producing an object file and a stand-alone executable.
  - https://en.wikipedia.org/wiki/Static_library
- Dinámico
  - It loads and links the shared libraries needed by an executable when it is executed (at "run time"), by copying the content of libraries from persistent storage to RAM, and filling jump tables and relocating pointers.
  - The specific operating system and executable format determine how the dynamic linker functions and how it is implemented.
  - https://en.wikipedia.org/wiki/Dynamic_linker

Introduction

Toolchain

Toolchain

Ligado

- Estático
  - Ventajas
    - Programas rápidos (no hacen operaciones extras al ejecutarse).
    - Se evitan problemas de dependencia (facilita distribución e instalación).
  - Desventajas
    - Programas más grandes.
- Dinámico
  - Ventajas
    - Programas más pequeños.
  - Desventajas
    - Los programas son más lentos.
    - Si hay cambios en la librería surgen incompatibilidades y hay que tener varias verisones.

Introduction

Toolchain

Mini-example on Linux

- Como se construye una aplicación que usa librerias dinámicas?
- Es el compilador (framework) que decide?
- Es el programdor que controla las cosas para elegir (estático/dinámico) lo que más le conviene según la aplicación?

Tarea

Hacer un a librería dinámica en Linux.

Veremos como se hace una libreia estática en unos slides más adelante.

Introduction

Bibliografía

Links

- Ligado
  - Dynamic Linking in Linux and Windows, part one
    - http://www.symantec.com/connect/articles/dynamic-linking-linux-and-windows-part-one
- Manuales
  - http://www.unix.com/man-page/osx/1/otool/

GNU

GNU is not UNIX

Gcc

GNU toolchain

Componentes

Projects included in the GNU toolchain are:

1. GNU make: an automation tool for compilation and build
2. GNU Compiler Collection (GCC): a suite of compilers for several programming languages
3. GNU Binutils: a suite of tools including linker, assembler and other tools
4. GNU Bison: a parser generator, often used with the Flex lexical analyser. YACC/Lex
5. GNU m4: an m4 macro processor.
6. GNU Debugger (GDB): a code debugging tool.
7. GNU build system (autotools): Autoconf, Automake and Libtool

GNU Binutils

Lista

1. **ld** - the GNU linker.
2. **as** - the GNU assembler.
3. addr2line - Converts addresses into filenames and line numbers.
4. ar - A utility for creating, modifying and extracting from archives.
5. c++filt - Filter to demangle encoded C++ symbols.
6. dlltool - Creates files for building and using DLLs.
7. gold - A new, faster, ELF only linker, still in beta test.
8. gprof - Displays profiling information.
9. nlmconv - Converts object code into an NLM.
10. nm - Lists symbols from object files.
11. objcopy - Copies and translates object files.
12. objdump - Displays information from object files.
13. ranlib - Generates an index to the contents of an archive.
14. readelf - Displays information from any ELF format object file.
15. size - Lists the section sizes of an object or archive file.
16. strings - Lists printable strings from files.
17. strip - Discards symbols.
18. windmc - A Windows compatible message compiler.
19. windres - A compiler for Windows resource files.

GNU port

Versiones exportadas a otras plataformas

MinGW (Minimalist GNU for Windows)

- MinGW provides a complete Open Source programming tool set which is suitable for the development of native MS-Windows applications, and which do not depend on any 3rd-party C-Runtime DLLs.
  - It does depend on a number of DLLs provided by Microsoft themselves, as components of the operating system; most notable among these is MSVCRT.DLL, the Microsoft C runtime library. Additionally, threaded applications must ship with a freely distributable thread support DLL, provided as part of MinGW itself.
- MinGW compilers provide access to the functionality of the Microsoft C runtime and some language-specific runtimes. MinGW, being Minimalist, does not, and never will, attempt to provide a POSIX runtime environment for POSIX application deployment on MS-Windows.
- If you want POSIX application deployment on this platform, please consider **Cygwin** instead.
- Primarily intended for use by developers working on the native MS-Windows platform, but also available for cross-hosted use.
- MinGW includes: A port of the GNU Compiler Collection (GCC), including C, C++, ADA and Fortran compilers; GNU Binutils for Windows (assembler, linker, archive manager).
- http://www.mingw.org/

GNU port

Versiones exportadas a otras plataformas

Cygwin

- Es una colección de herramientas desarrollada por Cygnus Solutions para proporcionar un comportamiento similar a los sistemas Unix en Microsoft Windows.
- Su objetivo es portar software que ejecuta en sistemas POSIX a Windows con una recompilación a partir de sus fuentes.
- Aunque los programas portados funcionan en todas las versiones de Windows, su comportamiento es mejor en Windows NT, Windows XP y Windows Server 2003.
- El sistema Cygwin tiene varias partes diferenciadas:
  - Una biblioteca de enlace dinámico («cygwin1.dll») que implementa la interfaz de programación de aplicaciones POSIX usando para ello llamadas a la API nativa de Windows.
  - Una cadena de desarrollo GNU (que incluye entre otras utilidades GCC y GDB) para facilitar las tareas básicas de desarrollo.
  - Aplicaciones equivalentes a los programas más comunes de los sistemas UNIX. Incluso, cuenta con un sistema X (Cygwin/X) desde 2001.

Compiling

Secuencia de pasos

Creación estática de bibliotecas

ar	Crea una biblioteca estática a partir de archivos objeto.
ranlib	Agrega o actualiza la tabla del contenido de una biblioteca.
nm	Muestra el contenido de una biblioteca (tabla de símbolos).

Ejemplo

```
$ gcc -c code.c
$ ar rc libNAME.a  archivos.o code.o
$ ranlib libNAME.a
$ nm libNAME.a
  code.o
  00000000  U _printf
     ...
```

Bibliotecas dinámica

Linux

- Son bibliotecas que se cargan en la memoria sólo cuando se invocan.
- Como se sabe:
  - Cuales han sido cargadas?
  - Cuanto espacio ocupan?
  - Cuando se descargan?
  - Como mejoro el comportamiento cargando y decargando manualmente?

GNU

Comandos

Explicación de cada uno

Gcc (GNU project C and C++ compiler)

Compilador

- GNU project C and C++ compiler. It includes front ends for Objective-C, Fortran, Java, Ada, and Go.
- When you invoke GCC it normally does prepocesing, compilation, assembly and linking.
- Web page: https://gcc.gnu.org/

Versiones
4.9, **5.5**, **6.4, 7.2**
10.2 (2020-07-23)

Uso

$ gcc [options –version] files
gcc 4.8.5 20150623 (RedHat 4.8.5.4) # Para un CentOS
Copyright @ 2015 …
This is free software ..

Apple LLVM version 7.3.0 (clang-703.0.31) # Para una MacOSX
Target: x86_64-apple-darwin15.6.0
Thread model: posix

Ld (Linker)

Ligador/Enlazador

Liga las librerias necesarias buscandolas en:

/usr/local/lib /lib

/usr/lib

En MacOSX /Library/Framework

$ ld –o file.exe file1.o file2.o /lib/crt.o

Temas relacionados con el ligado y la seguridad

Hooking

Variables de ambiente (las más "importantes")

- LD_LIBRARY_PATH
  - A list of directories in which to search for ELF libraries at execution-time.
- LD_PRELOAD
  - A list of additional, user-specified, ELF shared objects to be loaded before all others.

Ldd (List Dynamic Dependecies)

Ver que librerias se usan en un ejecutable

- Print shared object dependencies in Linux.

Uso

```
$ ldd executable
$ ldd /bin/ls
   linux-vdso-so.1 => (0x00…)
   /lib64/ld-linux-x86-64.so.2 (0x…)
   libpthread.so.0 => /lib64/libpthread.so.0 (0x…)
```

nm (symbol table)

Lista los símbolos de archivos objeto

- Lista los símbolos de un archivo.
- Para cada símbolo muestra:
  - El valor del símbolo
  - Tipo: B, b, D, d, i, I, U, etc.
  - Nombre: _exit, _ _DefaultRune , _ _ _error.

Uso

```
$ nm file
 1234567890123456 B _ _bss_start
 0000000000060100 T _init
 Valor del símbolo     Tipo de símbolo   Nombre del símbolo
```

objdump

Volcado de archivo objeto

- It is a program for displaying various information about object files.
- For instance, it can be used as a disassembler to view an executable in assembly form.

Uso

$ objdump [options] file
$ objdump –d /bin/ls # desensambla (genera ensamblador) del comando ls

Disassembly of section .init:

Dirección Opcode Ensamblador (nemónico + registros/valores)
402228: 48 83 ec 08 sub %0x8,%rsp
40222c: 48 8b 05 95 8d 21 00 mov 0x218d95(%rip),%rax

objdump

Volcado de archivo objeto

Ejemplos

$ objdump –D –d binary –m i8086 mbr.bin

Des-ensambla un archive binario

objdump

Volcado de archivo objeto

For MacOSX you use otool utility

$ otool -V -s <section> <file>
Disassemble a specific segment and a segment, e.g.,
$ otool -V -s __text __TEXT /bin/ls (= objdump -d /bin/ls)

$ otool -L <file>
Print out shared library dependencies (= ldd /bin/ls)

$ otool -tV <file>
Disassemble the text section (= objdump -j .text -d /bin/ls)

$ otool -dv <file>
Dump the contents of the data section (= objdump -j .data -s /bin/ls)

Patch

Parche

- You will distribute your code changes in patches and receive code from others as patches.
- *Incremental patches* provide an easy way to move from one kernel tree to the next.
- Instead of downloading each large tarball of the kernel source, you can simply apply an incremen- tal patch to go from one version to the next.
- This saves everyone bandwidth and you time.

Uso

To apply an incremental patch, from *inside* your kernel source tree, simply run:

```
$ patch –p1 < ../patch-x.y.z
```

GDB (GNU Project debugger)

Depurador / Debuger

- It allows you to see what is going on **inside** another program while it executes or what another program was doing at the moment it crashed.
- It can do four main kinds of things (plus other things in support of these) to help you catch bugs in the act:
  - Start your program, specifying anything that might affect its behavior.
  - Make your program stop on specified conditions (breakpoints).
  - Examine what has happened, when your program has stopped.
  - Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.
- The program being debugged can be written in Ada, C, C++, Objective-C, Pascal (and many other languages).
- Those programs might be executing on the same machine as GDB (native) or on another machine (remote). GDB can run on most popular UNIX and Microsoft Windows variants.
- The current version of GDB is 7.11.1.

GDB (GNU Project debugger)

Depurador / Debuger

- The debugger does not contain its own GUI, and defaults to a CLI (Command-Line Interface).
- Several front-ends have been built for it.

Links

- Tutoriales
  - http://davis.lbl.gov/Manuals/GDB/gdb.html
  - http://www.delorie.com/gnu/docs/gdb/gdb.html
  - http://www.tutorialspoint.com/gnu_debugger/
  - http://www.yolinux.com/TUTORIALS/GDB-Commands.html

GDB (GNU Project debugger)

Depurador / Debuger

GDB es una herramienta bastante compleja ya que permite ejecuciones via remota y con un módulo servidor.

- Command line arguments
- General command (help)
- Break and watch
- Line execution
- Stack
- Source code
- Machine language
- Examine variables
- GDB modes
- Start and stop

GNU labo

Prácticas

Realizar las siguientes prácticas básicas:

1. Crear un programa en C y depurarlo con GDB realizando:
    i. Colocar break points, ejecutar el código hasta uno, etc.
    ii. Inspeccionar el valor de variables y de variables tipo apuntador.
2. Crear una librería:
    i. Estática
    ii. Dinámica
3. Inspeccionar archivos binarios (ejecutables):
    i. Analizar la tabla de símbolos.
    ii. Identificar las librerias que utiliza.

LLVM

Compiler Infrastructure

Project