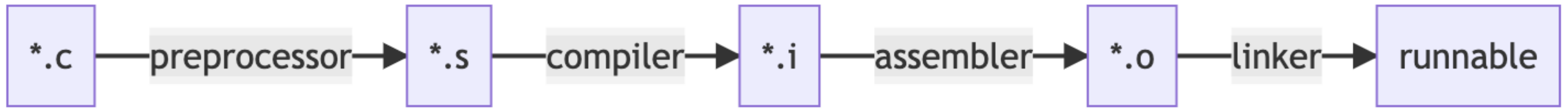
Lab2: C programing under linux

缪岱烨

GNU Compiler Collection(GCC)

The GNU Compiler Collection, commonly known as GCC, is a set of compilers and development tools available for Linux, Windows, various BSDs, and a wide assortment of other operating systems.



Preprocessor

function

A C Preprocessor is just a text substitution tool and it instructs the compiler to do required pre-processing before the actual compilation.

All preprocessor commands begin with a hash symbol (#). such as

```
#define //userd to define a macro
#include //Used to include a file in the source code program
#ifdef  //Used to include a section of code if a certain macro is defined by #define
#ifndef //contrary to #ifdef
#endif //Used to mark the end of #if, #ifdef, and #ifndef
```

command

```
gcc -E *.c -o *.i
```

Compiler

defination

Compilation is the process the computer takes to convert a high-level programming language into a machine language that the computer can understand.

The software which performs this conversion is called a compiler.

command

```
gcc -S *.i -o *.s
```

Assembler

defination

An assembler is a type of computer program that interprets software programs written in assembly language into machine language, code and instructions that can be executed by a computer.

command

```
gcc -c *.s -o *.o
```

Linker

defination

In computing, a linker or link editor is a computer system program that takes one or more object files (generated by a compiler or an assembler) and combines them into a single executable file, library file, or another "object" file.

create your own library

create static library

```
ar -cr lib***.a *.o *.o
```

create dynamic library

```
gcc file1 file2 -shared -o lib***.so
```

link path and and releated method

manually sepcify

```
gcc -Iheader_file_directory -Llibrary_directory
```

environment variable

```
export LIBRARY_PATH=dir:$LD_LIBRARY
export LD_LIBRARY_PATH=dir:$LD_LIBRARY_PATH
```

Default search path

```
mv library Default_search_path_directory
```

GDB: The GNU Project Debugger

GDB, the GNU Project debugger, 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.

GDB can do four main kinds of things

- Start your program.
- Make your program stop on specified conditions.
- 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

Run C Using GNU symbolic debugger(GDB)

Command	Remark
gcc -g file	generate a debuggable file
gdb file	start debug runnable file
run/r	start running file

set breakpoint

Command	Parameters	Remark
break/b	numline function offset filename:numline filename:function	set breakpoint
watch/rwatch/awatch	expr	set watchpoint
info	breakpoints/watchpoints	check breakpoints infomation

Remove breakpoint

Command	Parameters	Remark
clear/c	same as break	remove breakpoints
delete	braekpoints number	base on breakpoint number delete it
enable/disable	breakpoints number	control open/close breakpoints without delete

Continue to operate

Command	Remark
continue/c	Resume program execution
next/n	continue to the next source line
step/s	enters a function
until/u	continue running until the specified source

other

Command	Parameters	Remark
backtarce/bt	n	check all stack information
print/p	args parameters_name	check parameters inforamtion
list	linenum function offset filename:linenum	list file content