

实验引导 0 - 汇编相关工具

TAs

- 杨耀良
- 罗开荣
- 刘晓义
- 王文新

请多使用网络学堂的课程邮件功能

TAs

- 杨耀良
- 罗开荣
- 刘晓义
- 王文新

请多使用网络学堂的课程邮件功能

TL;DR

- 环境配置
- 反汇编器的基本使用
- 调试器的基本使用

环境配置

本课程的实验均在服务器集群上进行 (x86-64, Linux), 请安装 SSH 客户端:

- Ubuntu: `apt install openssh-client`
- Windows & macOS: 在终端中已内置

```
$ ssh <username>@<server IP>
```

在本地安装环境

- 在 Windows 下推荐安装 WSL¹，发行版使用 Ubuntu。
- 在 macOS 下推荐使用 VirtualBox² 安装 Ubuntu 虚拟机。

¹<https://learn.microsoft.com/zh-cn/windows/wsl/install>

²<https://www.virtualbox.org/wiki/Downloads>, 请注意处理器类型, M[1-4] 为 Apple Silicon Host

在本地安装环境

- 在 Windows 下推荐安装 WSL³，发行版使用 Ubuntu。
- 在 macOS 下推荐使用 VirtualBox⁴ 安装 Ubuntu 虚拟机。

在 Ubuntu 内安装需要的工具：

```
$ sudo apt install build-essential gcc-multilib \
g++-multilib gdb binutils
```

³<https://learn.microsoft.com/zh-cn/windows/wsl/install>

⁴<https://www.virtualbox.org/wiki/Downloads>, 请注意处理器类型, M[1-4] 为 Apple Silicon Host

使用编辑器

推荐使用 VSCode，可以方便地远程连接服务器、WSL 和虚拟机，并且提供了一个 gdb 前端。

- 下载: <https://code.visualstudio.com/download>
- 推荐安装以下插件:
 - C / C++ Extension Pack
 - GDB Debug (By DamianKoper)
 - Remote Development

编译器 gcc 选项

回忆 gcc 的普通用法:

```
$ gcc -o executable source.c
```

```
$ gcc -O3 -o executable source.c
```


编译器 gcc 选项

回忆 gcc 的普通用法:

```
$ gcc -o executable source.c
```

```
$ gcc -O3 -o executable source.c
```

本课程中可能会用到以下**新**的编译器选项:

- -g: 添加调试信息
- -S: 输出汇编代码

编译器 gcc 选项

回忆 gcc 的普通用法:

```
$ gcc -o executable source.c
```

```
$ gcc -O3 -o executable source.c
```

本课程中可能会用到以下**新**的编译器选项:

- -g: 添加调试信息
- -S: 输出汇编代码

g++ 也有相同的选项

编译带有调试信息的可执行文件

```
$ gcc -g -o executable source.c
```

编译带有调试信息的，优化过的可执行文件

```
$ gcc -g -O3 -o executable source.c
```

编译到汇编

```
$ gcc -S -o asm.S source.c
```

反汇编器 objdump

objdump 可以将二进制可执行文件中的指令翻译回文本汇编指令：

```
$ objdump -D ./executable
```

- -d: 只反汇编可执行代码段
- -D: 反汇编所有段
- -s: 添加源码信息

反汇编器 objdump

objdump 可以将二进制可执行文件中的指令翻译回文本汇编指令：

```
$ objdump -D ./executable
```

- -d: 只反汇编可执行代码段
- -D: 反汇编所有段
- -s: 添加源码信息

一些其他选项包括：

- --disassembler-color=on: 打开高亮输出
- --visualize-jumps=color: 显示跳转目标

其他工具

- `file`: 可以快速查看文件类型。当文件为可执行文件时，回输出架构、链接形式等简要信息。
- `readelf`: 查看 ELF 中的各类细节信息。
- `less`: 在终端中对大段文字进行滚动、搜索。

反汇编到文件

```
$ objdump -D executable > disasm.S
```

反汇编代码段，添加源码信息，并查看

```
$ objdump -S executable --visualize-jumps=color \  
--disassembler-color=on | less -R
```

查看可执行文件格式

```
$ file executable
```

查看全部 elf 信息

```
$ readelf -a executable | less
```

调试器 GDB

GDB 可以单步调试可执行程序

- 设置断点：在设定点停止程序
- 观察变量值、寄存器值变化
- 路径跟踪：[单/多]步 执行 指令或 c 代码语句，观察行为
- 错误查找：自动停在 abort 处

```
$ gdb ./executable
```


调试基本流程

GDB 调试的总体流程为：

- 针对感兴趣的程序状态设置断点，例如进入特定函数、执行到某一行，某个变量被存取，特定条件满足等。
- 执行程序。经过一段时间，程序抵达断点，执行被暂停。
- 检查变量、执行栈、寄存器堆等状态，使用步进逐步执行程序。
- 从暂停状态中恢复执行。

调试基本流程

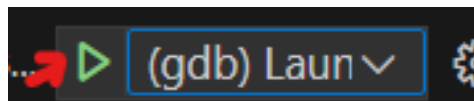
GDB 调试的总体流程为：

- 针对感兴趣的程序状态设置断点，例如进入特定函数、执行到某一行，某个变量被存取，特定条件满足等。
- 执行程序。经过一段时间，程序抵达断点，执行被暂停。
- 检查变量、执行栈、寄存器堆等状态，使用步进逐步执行程序。
- 从暂停状态中恢复执行。

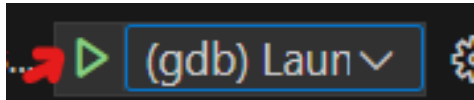
或者：程序 Segment fault 了，需要看看怎么挂的。

VSCode 配置

- 在侧边栏 → Run and Debug: **create a launch.json file**
→ **C++ (GDB/LLDB)**
- 在打开的编辑器内: **Add Configuration...** → **C/C++: GDB (Launch)**
- 修改 program 为编译得到的二进制文件路径, 例如
`${workspaceFolder}/executable`



VSCode 配置

- 在侧边栏 → Run and Debug: **create a launch.json file**
→ **C++ (GDB/LLDB)**
- 在打开的编辑器内: **Add Configuration...** → **C/C++: GDB (Launch)**
- 修改 program 为编译得到的二进制文件路径, 例如
`${workspaceFolder}/executable`
- 
- 如果闪退了, 请检查编译时有没有添加 `-g` 选项 (调试信息)
- `Ctrl-P` → `Open Disassembly View` 可以在汇编模式下调试

VSCode GDB 交互

- 使用浮动窗口进行暂停、恢复等执行流控制。
 - Step Over: 执行到下一行完成
 - Step Into: 如果下一行中有函数，进入函数内
 - Step Out: 执行至当前函数返回
- 点击行号左侧添加断点，右键点击局部变量可以添加写入断点。
- 在 Watch 窗口中可以添加任意表达式进行求值

GDB 终端命令

执行流控制

- `run [参数]`: 启动程序, 可以提供命令参数
- `c`: 在暂停状态时继续执行
- `n [步数]`: 步进
- `s [步数]`: 步入 (进入函数)
- `finish`: 执行至函数完成。
- `si/ni [步数]`: 指令版本的 `s/n`
- `q`: 退出

打印信息

- `p <变量/$rax/表达式>`: 打印变量、寄存器或者表达式的值。

- `p/x <变量/$rax/表达式>`: 以 16 进制格式打印。
- `bt`: 显示调用栈
- `info reg`: 显示寄存器堆

断点相关

- `break <函数名/文件:行数/*addr> [if 表达式]`: 设置断点。如果有条件，只有当条件表达式为真时暂停执行。
- `watch <变量/表达式> [if 表达式]`: 设置 Watchpoint, 当变量值发生变化时暂停，同样可以带有条件。
- `rwatch <变量/表达式> [if 表达式]`: 侦测读取的 watch 版本。
- `info break/watch`: 查看断点和 watchpoint 信息。
- `d <ID>`: 删除断点 / watchpoint

界面

- `layout asm/src/split`: 修改界面为源码、汇编或分屏。

启动调试器

```
$ gdb ./executable
```

```
(gdb) layout split
```

```
(gdb) b foobar
```

```
(gdb) run
```

```
(gdb) # GDB 的命令非常多，可以多求救：
```

```
(gdb) help
```


Q&A