准备工作

- 准备工作
  - 编译带有 Debug 信息的 Glibc

编译带有 Debug 信息的 Glibc

在 Ubuntu 下想要调试 Glibc 源码有很多方法，可以通过 apt 下载 Glibc 相关的编译信息，配合指定源码目录可以实现调试，缺点是无法尝试修改源码验证学习，因此本文采用编译出 Glibc 源码的方案。

```
# GDB 命令：set substitute-path  /build/glibc-eX1tMB/glibc-2.31
/home/layton/Tools/glibc-2.31

"configurations": [
    {
        "name": "Debug",
        "type": "gdb",
        "request": "launch",
        "target":
"/home/layton/WorkSpace/GLibcDebug/build/test/automatedtools_unittests",
        "cwd": "${workspaceRoot}",
        "valuesFormatting": "parseText",
        # 如下 通过 Substitutions 制定映射源码路径
        "pathSubstitutions": {
            "/build/glibc-eX1tMB/glibc-2.31": "/home/layton/Tools/glibc-
2.31"
        }
    }
]
```

编译 Glibc 源码：

1. 下载源码：sudo apt source glibc-source;
2. 解压并创建编译目录，与源码目录同级;

```
❯ ls glibc-* -d
glibc-2.31  glibc-debug
```

3. 进入编译目录进行编译，2.31 源码如果不启用 --enable-obsolete-rpc 会在 install 时缺少 xdr_* 报错失败;

```
cd glibc-debug

../glibc-2.31/configure --prefix=/home/layton/Tools/glibc-debug/usr --
enable-obsolete-rpc --enable-obsolete-nsl --enable-debug=yes CFLAGS="-
O2 -g" CPPFLAGS="-O2 -g"
```

```
mkdir usr && make all && make install
```

4. 工程中使用 cmake 指定 link_dir;

```
link_directories(/home/layton/Tools/glibc-debug/usr/lib)
```

5. 编译完成后使用 ldd 命令查看动态库;

```
❯ ldd build/test/automatedtools_unittests
        linux-vdso.so.1 (0x00007ffdf7974000)
        libpthread.so.0 => /home/layton/Tools/glibc-
debug/usr/lib/libpthread.so.0 (0x00007fceeca3b000)
        libstdc++.so.6 => /lib/x86_64-linux-gnu/libstdc++.so.6
(0x00007fceec842000)
        libm.so.6 => /home/layton/Tools/glibc-debug/usr/lib/libm.so.6
(0x00007fceec6fd000)
        libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1
(0x00007fceec6e2000)
        libc.so.6 => /home/layton/Tools/glibc-debug/usr/lib/libc.so.6
(0x00007fceec51f000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fceeca5f000)
```