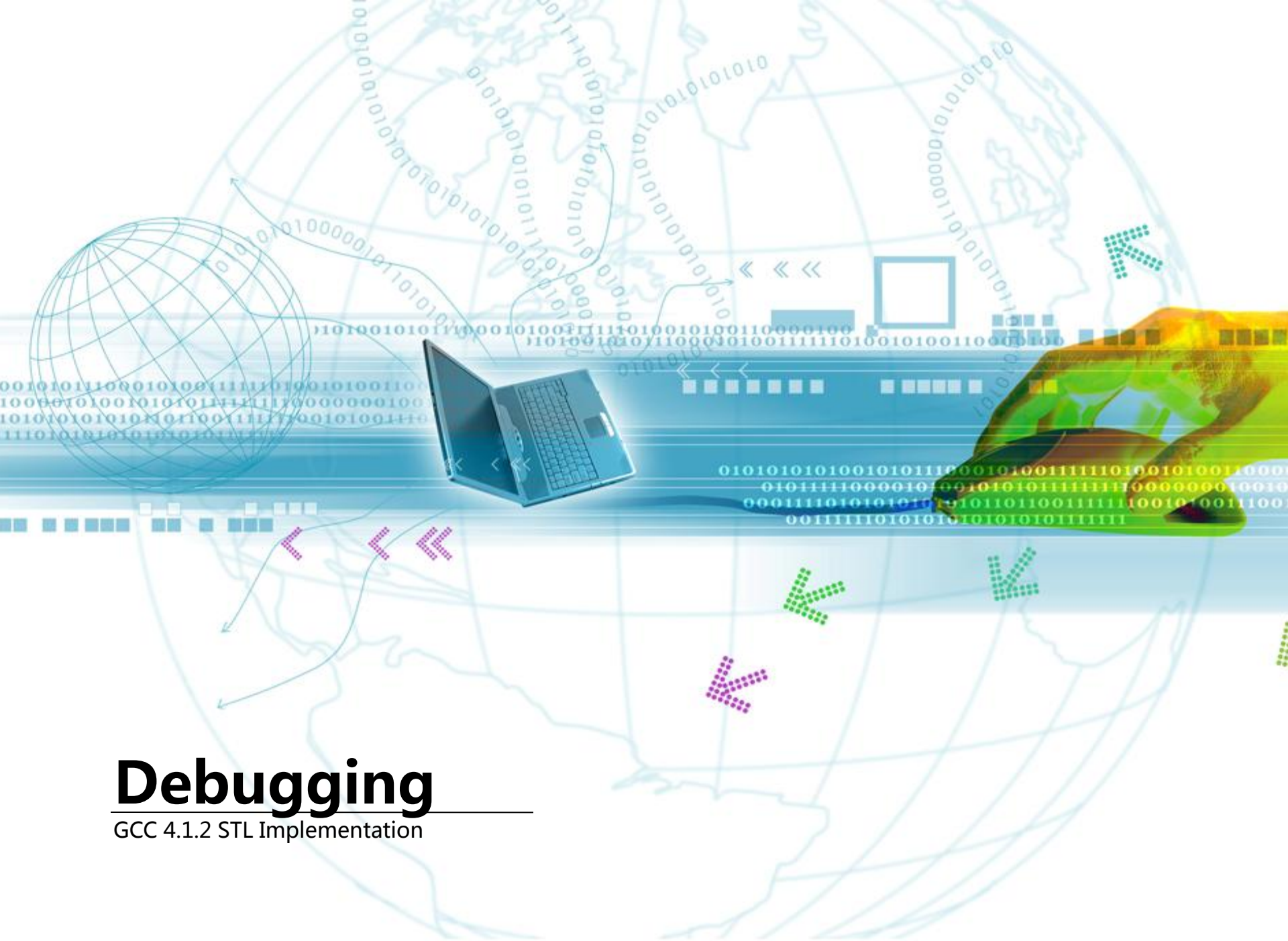

Debugging

GCC 4.1.2 STL Implementation

GDB PYTHON

- 确认GDB是否支持PYTHON
- 用python实现gdb命令
- 实现*xmlNode*的pretty-printer
- pretty-printer工作原理
- 实现自己的pretty-printer

确认GDB是否支持PYTHON

打开gdb,执行python help(gdb)查看运行结果
如果出现以下字样则不支持：

Scripting in the "Python" language is not supported in this copy of GDB

解决方案:

1. 安装python对应版本的python-dev
2. gdb必须为7.0以上版本
3. ./configure –with-python;

   make

   make install

用python实现gdb命令

```
import gdb

class SaveBreakpointsCommand(gdb.Command):
    """Save the current breakpoints to a file.
    This command takes a single argument, a file name.
    The breakpoints can be restored using the 'source' command."""

    def __init__(self):
        super(SaveBreakpointsCommand, self).__init__("save breakpoints",
                                                     gdb.COMMAND_SUPPORT,
                                                     gdb.COMPLETE_FILENAME)

    def invoke(self, arg, from_tty):
        f = open(arg, 'w')
        for bp in gdb.breakpoints():
            print >> f, "break", bp.location,

SaveBreakpointsCommand()
```

用python实现gdb命令

- 对python gdb中所有的类都可以通过python help(xxx)查看用法
- init函数第一个参数定义了命令名，最后一个参数表示可以使用自动补全
- invoke是命令实现的地方
- gdb.breakpoints()返回所有的断点。每个断点用一个Breakpoint对象表示
- location是Breakpoint对象的属性，表示打断点的位置

加载gdb脚本

- source script-path
  - 加载gdb脚本或python脚本
  - 通过.gdbinit自动加载
- program-gdb.py
  - 自动加载带有-gdb.py的脚本
- add-auto-load-safe-path path
  - 将路径添加到加载路径中

练习

- 实现一个命令，可以打印出当前函数的名字

实现xmlNode的pretty-printer

- 什么是pretty-printer

```
(gdb) p s
$3 = {static npos = 18446744073709551615, _M_dataplus = {<
std::allocator<char>> = {<__gnu_cxx::new_allocator<char>>
= {<No data fields>}, <No data fields>},
    _M_p = 0x502028 "0123456789"}}
(gdb)
```

```
(gdb) p s
$1 = 0123456789
(gdb)
```

实现string的pretty-printer

- 实现一个printer类

```
class StdStringPrinter:
    def __init__(self, val):
        self.val = val

    def to_string(self):
        return self.val['_M_dataplus']['_M_p'].string()
```

- 注册pretty-printer

```
def build_pretty_printer():
    pp = gdb.printing.RegexpCollectionPrettyPrinter("my_library")
    pp.add_printer("string_printer", "^std::basic_string<char,.*>$", StdStringPrinter)
    return pp

gdb.printing.register_pretty_printer(gdb.current_objfile(), build_pretty_printer())
```

练习

- 显示xmlNode父节点名字

pretty-printer工作原理

- gdb内置实现了python gdb模块
- gdb模块将当前调试的程序的信息包装成Frame、Thread、Block、Breakpoints传递给python解释器
- Python解释器返回给gdb解析的结果

实现自己的pretty-printer

```c
#include <stdlib.h>

typedef struct
{
    int *p;
    int len;
}SubStruct;

typedef struct
{
    int length;
    SubStruct* data;
}Sample;

void createSubStruct(SubStruct* subStruct);
Sample* createSample(int len);

int main() {
    srand(9527);
    Sample* s = createSample(3);
    return 0;
}
```

- 实现Sample的Pretty-Printer
- 实现SubStruct的Pretty-Printer
- 在main函数中下断点, 加载Pretty-Printer输出s的内容

Q & A

Thank You