



中国海洋大学

系统开发工具实验报告



所在学院：信息科学与工程学部 专 业：计算机科学与技术

学 生 姓 名：陈一韬 学 号：23010022003

2024/8/27

主 题：调试及性能分析、元编程、PyTorch 编程以及“大杂烩”

1 实验目的

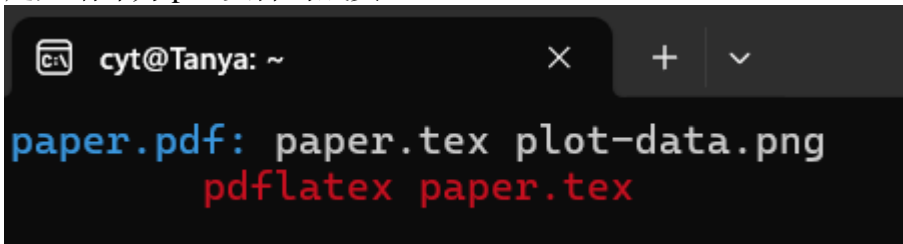
1. 学习调试与性能分析的基本工具与方法
2. 学习使用 makefile 方式进行元编程
3. 学习使用 PyTorch 进行简单的程序开发
4. 学习“大杂烩”模块内相关知识

2 课后实例练习

元编程部分

1. paper.pdf: paper.tex plot-data.png

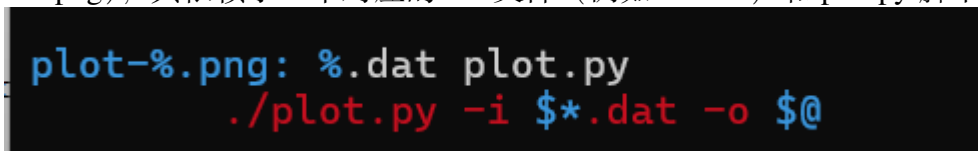
定义编译为 pdf 文件的成员



```
cyt@Tanya: ~  
paper.pdf: paper.tex plot-data.png  
    pdflatex paper.tex
```

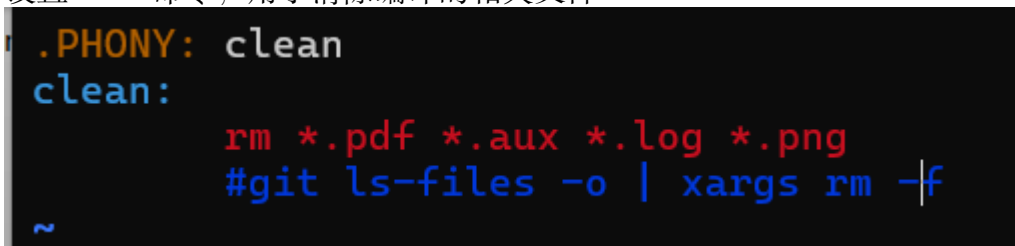
2. plot-%.png: %.dat plot.py

模式规则 (pattern rule)，用于生成文件名以 plot- 开头且扩展名为.png 的文件（例如 plot-foo.png），其依赖于一个对应的.dat 文件（例如 foo.dat）和 plot.py 脚本。



```
plot-%.png: %.dat plot.py  
    ./plot.py -i $*.dat -o $@
```

3. 设置 clean 命令，用于清除编译的相关文件



```
.PHONY: clean  
clean:  
    rm *.pdf *.aux *.log *.png  
    #git ls-files -o | xargs rm -f  
~
```

4. make 指令

自动化构建工具，它的主要作用是管理和控制程序的构建过程。它通过读取一个叫做 Makefile 的配置文件来确定如何生成目标文件。

```
cyt@Tanya:~$ make
pdflatex paper.tex
This is pdfTeX, Version 3.141592653-2.6-1.40.22 (TeX Live 2021)
restricted \write18 enabled.
entering extended mode
(./paper.tex
LaTeX2e <2021-11-15> patch level 1
L3 programming layer <2022-01-21>)
*
```

5. make clean

调用 clean 指令，清除相关文件

```
cyt@Tanya:~$ make clean
rm *.pdf *.aux *.log *.png
rm: cannot remove '*.pdf': No such file or directory
rm: cannot remove '*.aux': No such file or directory
make: *** [makefile:9: clean] Error 1
```

调试与性能分析

1. @profile

在函数前设置装饰器，以便 line-profiler 对函数进行性能分析

```
@profile
def insertionsort(array):

    for i in range(len(array)):
        j = i-1
        v = array[i]
        while j >= 0 and v < array[j]:
```

2. python -m cProfile -s time sorts.py

用于运行 Python 脚本 sorts.py 并使用 cProfile 模块进行性能分析。

```
PS C:\Users\xiaohei\Downloads> python -m cProfile -s time s
1298 function calls (1270 primitive calls) in 0.00
```

Ordered by: internal time

| ncalls | totttime | percall | cumtime | percall | filename:line |
|--------|----------|---------|---------|---------|---------------|
| 12 | 0.001 | 0.000 | 0.001 | 0.000 | {built-in met |
| 2 | 0.001 | 0.000 | 0.001 | 0.000 | {built-in met |
| 6/1 | 0.000 | 0.000 | 0.004 | 0.004 | <frozen impor |
| 40 | 0.000 | 0.000 | 0.000 | 0.000 | <frozen impor |
| 2 | 0.000 | 0.000 | 0.000 | 0.000 | {built-in met |
| 2 | 0.000 | 0.000 | 0.000 | 0.000 | {method 'read |
| 2 | 0.000 | 0.000 | 0.000 | 0.000 | {method '__ex |
| 8 | 0.000 | 0.000 | 0.001 | 0.000 | <frozen impor |
| 6 | 0.000 | 0.000 | 0.001 | 0.000 | <frozen impor |
| 4 | 0.000 | 0.000 | 0.000 | 0.000 | {built-in met |
| 1 | 0.000 | 0.000 | 0.002 | 0.002 | random.py:1(< |

3. kernprof -l -v sorts.py 用于在 Python 脚本 sorts.py 中执行性能分析，特别是使用 line_profiler 进行逐行性能分析。

```

PS C:\Users\xiaohei\Downloads> kernprof -l -v sorts.py
Wrote profile results to sorts.py.lprof
Timer unit: 1e-06 s

Total time: 0.201388 s
File: sorts.py
Function: insertionsort at line 10

Line #      Hits          Time    Per Hit   % Time  Line Content
=====
    10                               @profile
    11                               def insert
    12
    13      25911       7059.6         0.3     3.5      for i
    14      24911       6096.2         0.2     3.0      j
    15      24911       6527.4         0.3     3.2      v
    16     225874      72003.4         0.3    35.8      wh
    17     200963      53435.2         0.3    26.5
    18     200963      49336.6         0.2    24.5
    19      24911       6446.5         0.3     3.2      a
    20      1000        482.9         0.5     0.2      retur

Total time: 0.133959 s
File: sorts.py
Function: quicksort at line 22

Line #      Hits          Time    Per Hit   % Time  Line Content
=====
    22                               @profile

```

4. journalctl | grep sudo

快速查看与 sudo 相关的操作和日志记录，非常适合调试和审计目的。它可以帮助用户了解在系统上执行 sudo 命令的历史记录和相关信息。

```

cyt@Tanya:~$ journalctl | grep sudo
Aug 30 11:31:18 Tanya usermod[427]: add 'cyt' to group
Aug 30 11:31:18 Tanya usermod[427]: add 'cyt' to shado
Aug 30 12:07:18 Tanya sudo[633]: pam_unix(sudo:auth):
Aug 30 12:07:18 Tanya sudo[633]: pam_unix(sudo:auth):
Sep 13 09:43:05 Tanya sudo[7084]:      cyt : TTY=pts/0
Sep 13 09:43:05 Tanya sudo[7084]: pam_unix(sudo:session)
Sep 13 09:43:26 Tanya sudo[7084]: pam_unix(sudo:session)
Sep 13 09:43:41 Tanya sudo[7647]:      cyt : TTY=pts/0
Sep 13 09:43:41 Tanya sudo[7647]: pam_unix(sudo:session)
Sep 13 09:43:59 Tanya sudo[7647]: pam_unix(sudo:session)
Sep 13 09:50:48 Tanya sudo[9582]:      cyt : TTY=pts/0
essential
Sep 13 09:50:48 Tanya sudo[9582]: pam_unix(sudo:session)
Sep 13 09:51:33 Tanya sudo[9582]: pam_unix(sudo:session)
Sep 13 09:54:35 Tanya sudo[11184]:      cyt : TTY=pts/0
ve-full
Sep 13 09:54:35 Tanya sudo[11184]: pam_unix(sudo:session)
Sep 13 09:56:01 Tanya sudo[11184]: pam_unix(sudo:session)
Sep 13 09:56:35 Tanya sudo[11707]:      cyt : TTY=pts/0

```

Pytorch 运用实例

1. torch.tensor(data)

创建一个张量。

X 为输入特征 (1.0, 2.0, 3.0, 4.0)。

y 为目标输出 (2.0, 3.0, 4.0, 5.0)，目标是希望模型能够学习到：输入加 1 即为输出。

```

# 1. 创建一个简单的数据集
# 特征
X = torch.tensor( data: [[1.0], [2.0], [3.0], [4.0]], dtype=torch.float32)
# 标签
y = torch.tensor( data: [[2.0], [3.0], [4.0], [5.0]], dtype=torch.float32)

```

2. class SimpleModel(nn.Module):

```
def __init__(self):
```

```
super(SimpleModel, self).__init__()
self.linear = nn.Linear(1, 1)
```

```
def forward(self, x):
    return self.linear(x)
```

2. 定义一个简单的线性回归模型

2 usages

```
class SimpleModel(nn.Module):
    def __init__(self):
        super(SimpleModel, self).__init__()
        self.linear = nn.Linear(in_features=1, out_features=1)

    def forward(self, x):
        return self.linear(x)
```

定义了一个名为 SimpleModel 的简单线性回归模型：

self.linear = nn.Linear(1, 1) 表示该模型有一个输入特征和一个输出特征。

forward 方法定义了如何通过模型进行前向传播。

3. criterion = nn.MSELoss()

使用均方误差损失 (MSELoss) 来度量模型的预测和实际输出之间的差异。

```
# 3. 初始化模型、损失函数和优化器
model = SimpleModel()
criterion = nn.MSELoss() # 均方误差损失
```

4. optimizer = optim.SGD(model.parameters(), lr=0.01)

随机梯度下降 (SGD) 优化器用于更新模型参数。

```
optimizer = optim.SGD(model.parameters(), lr=0.01) # 学习率为0.01
```

5. for epoch in range(100):

...

optimizer.step()

在训练过程中，每个 epoch 中我们会进行前向传播、计算损失、反向传播和参数更新。

使用 optimizer.zero_grad() 清除上一步的梯度，以避免累加。

```
# 4. 训练模型
for epoch in range(100): # 训练100个epochs
    model.train() # 将模型设置为训练模式
    optimizer.zero_grad() # 清空梯度
```

6. model.eval()

```
with torch.no_grad():
    test_input = torch.tensor([[5.0]], dtype=torch.float32)
    predicted = model(test_input)
    print(f' 预测值: {predicted.item():.4f}')
```

在训练完成后，我们把模型设置为评估模式 (eval)，并使用 `torch.no_grad()` 禁用梯度计算以提升性能。

```
# 5. 测试模型
model.eval() # 将模型设置为评估模式
with torch.no_grad():
    test_input = torch.tensor(data: [[5.0]], dtype=torch.float32)
    predicted = model(test_input)
    print(f'预测值: {predicted.item():.4f}')
```

Markdown 入门

在最后一章“大杂烩”中，列举了一系列有用的技能，其中我非常愿意深入学习 markdown 相关技术。因为其作为一种轻量级标记语言，无论是用来做笔记还是写报告都非常好用。可以说是兼具了 latex 的工整和 word 的所见即所得的特性。

1. 标题

使用 `#` 来创建标题，`#` 的数量表示标题的级别。

一级标题

二级标题

三级标题

2. 粗体和斜体

斜体使用 `*` 或 `_`：* 斜体文本 * 或 _ 斜体文本 _ 粗体使用 `**` 或 `__`：** 粗体文本 ** 或 __ 粗

体文本 __

粗体文本 *斜体文本* 下划线

C++ ▾

```
#include <iostream>
```

3. 代码

行内代码使用反引号 ‘`’:

这是 ‘行内代码’

块代码使用三个反引号或缩进四个空格:

C++ ▾

```
#include <iostream>
using namespace std;
int main ()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

4. 链接和图片

链接:

链接文本 (<http://example.com>)

图片:

! 替代文本 (<http://example.com/image.jpg>)

```
return 0;  
}
```



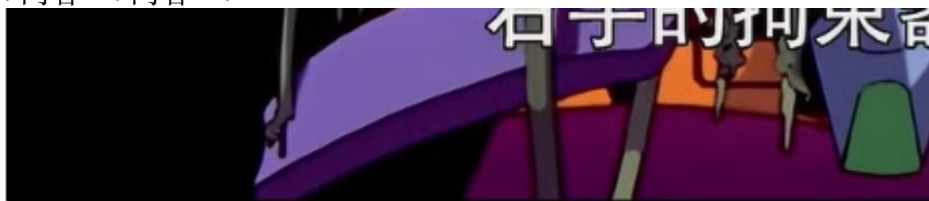
5. 表格

使用 | 来创建表格：

|列 1|列 2|

|—|—|

|内容 1|内容 2|



| | 列1 |
|----|--------------|
| 行1 | <u>item1</u> |
| 行2 | <u>item2</u> |

3 解题感悟

在这节系统开发工具基础课中，我对多种计算机相关技能都有了更深入的了解。

调试与性能分析

通过实际操作 `cprofiler` 和 `line-profiler`，我学会了如何有效地识别代码中的瓶颈。`cprofiler` 提供了函数级别的性能数据，而 `line-profiler` 则能更精确地分析代码中每一行的执行时间。这两种工具结合使用，可以帮助我更全面地评估代码的性能表现，并做出相应的优化。

元编程与 Makefile

使用 `makefile` 进行元编程的过程也极大地拓宽了我的视野。元编程不仅提高了代码的复用性，还简化了构建过程。通过掌握 `makefile`，我能够更高效地管理项目的构建系统，尤其是在处理大型项目时，这种工具的价值不言而喻。

PyTorch 模型编程

在使用 `PyTorch` 进行简单的模型编程时，我体会到了深度学习框架的强大与灵活。通过实际动手编写模型，我不仅加深了对深度学习基本概念的理解，还掌握了如何构建与训练模型的基本流程。这让我对未来研究与应用深度学习技术充满了期待。

Markdown 的应用

最后，Markdown 作为一种轻量级标记语言，我在课程中学习的使用方法，使我能够更方便地进行文档撰写与格式排版。良好的文档记录对于项目的推动和团队沟通至关重要，而 Markdown 的简洁性恰好符合了这一需求。

总结

总结来说，这次课程的学习让我收获颇丰。通过对调试工具、元编程、深度学习框架以及文档编写工具的深入了解与实践，我不仅提升了自己的技术技能，也增强了我对开发流程的整体理解。未来，我将继续在这些领域中探索与实践，不断提升自己的专业能力。

本次实验报告的 github 地址：<https://github.com/Shiny-Magikarp/->.git(相关.py 文件也已上传)