



中国海洋大学

系统开发工具基础实验报告

上课时间:	周五 1-4 节
姓 名:	张仕达
学 号:	23010022094
指导老师:	周小伟

1 实验内容

1.1 主题 1：调试：获取最近一天中超级用户的登录信息及其所执行的指令

1.1.1 内容

使用 Linux 上的 `journalctl` 或 macOS 上的 `log show` 命令来获取最近一天中超级用户的登录信息及其所执行的指令。

如果找不到相关信息，您可以执行一些无害的命令，例如 `sudo ls` 然后再次查看。

1.1.2 结果

```
-- Logs begin at Wed 2024-09-11 11:53:31 CST, end at Wed 2024-09-11 15:22:51 CST. --
Sep 11 11:53:31 localhost.localdomain systemd-journal[87]: Runtime journal is using 6.0M (max allowe
Sep 11 11:53:31 localhost.localdomain kernel: Initializing cgroup subsys cpuset
Sep 11 11:53:31 localhost.localdomain kernel: Initializing cgroup subsys cpu
Sep 11 11:53:31 localhost.localdomain kernel: Initializing cgroup subsys cpuacct
Sep 11 11:53:31 localhost.localdomain kernel: Linux version 3.10.0-957.el7.x86_64 (mockbuild@kbuilde
Sep 11 11:53:31 localhost.localdomain kernel: Command line: BOOT_IMAGE=/vmlinuz-3.10.0-957.el7.x86_6
Sep 11 11:53:31 localhost.localdomain kernel: e820: BIOS-provided physical RAM map:
Sep 11 11:53:31 localhost.localdomain kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009ebff]
Sep 11 11:53:31 localhost.localdomain kernel: BIOS-e820: [mem 0x000000000009ec00-0x000000000009ffff]
Sep 11 11:53:31 localhost.localdomain kernel: BIOS-e820: [mem 0x00000000000dc000-0x00000000000ffff]
Sep 11 11:53:31 localhost.localdomain kernel: BIOS-e820: [mem 0x0000000000100000-0x00000000003fedffff]
Sep 11 11:53:31 localhost.localdomain kernel: BIOS-e820: [mem 0x00000000003fee0000-0x00000000003fefffff]
Sep 11 11:53:31 localhost.localdomain kernel: BIOS-e820: [mem 0x00000000003feff000-0x00000000003feffff]
Sep 11 11:53:31 localhost.localdomain kernel: BIOS-e820: [mem 0x00000000003fff0000-0x00000000003ffff]
Sep 11 11:53:31 localhost.localdomain kernel: BIOS-e820: [mem 0x0000000000f0000000-0x0000000000f7ffff]
Sep 11 11:53:31 localhost.localdomain kernel: BIOS-e820: [mem 0x000000000fec000000-0x000000000fec0ffff]
Sep 11 11:53:31 localhost.localdomain kernel: BIOS-e820: [mem 0x000000000fee000000-0x000000000fee00fff]
Sep 11 11:53:31 localhost.localdomain kernel: BIOS-e820: [mem 0x000000000ffe000000-0x000000000ffff]
Sep 11 11:53:31 localhost.localdomain kernel: NX (Execute Disable) protection: active
Sep 11 11:53:31 localhost.localdomain kernel: SMBIOS 2.7 present.
Sep 11 11:53:31 localhost.localdomain kernel: DMI: VMware, Inc. VMware Virtual Platform/440BX Deskto
Sep 11 11:53:31 localhost.localdomain kernel: Hypervisor detected: VMware
Sep 11 11:53:31 localhost.localdomain kernel: vmware: TSC freq read from hypervisor : 3992.500 MHz
Sep 11 11:53:31 localhost.localdomain kernel: vmware: Host bus clock speed read from hypervisor : 66
Sep 11 11:53:31 localhost.localdomain kernel: vmware: using sched offset of 6712278988 ns
Sep 11 11:53:31 localhost.localdomain kernel: e820: update [mem 0x00000000-0x000000fff] usable ==> re
Sep 11 11:53:31 localhost.localdomain kernel: e820: remove [mem 0x000a0000-0x000ffff] usable
Sep 11 11:53:31 localhost.localdomain kernel: e820: last_pfn = 0x40000 max_arch_pfn = 0x40000000
Sep 11 11:53:31 localhost.localdomain kernel: MTRR default type: uncachable
Sep 11 11:53:31 localhost.localdomain kernel: MTRR fixed ranges enabled:
Sep 11 11:53:31 localhost.localdomain kernel: 00000-9ffff write-back
Sep 11 11:53:31 localhost.localdomain kernel: a0000-bffff uncachable
Sep 11 11:53:31 localhost.localdomain kernel: c0000-cbfff write-protect
Sep 11 11:53:31 localhost.localdomain kernel: cc000-effff uncachable
Sep 11 11:53:31 localhost.localdomain kernel: f0000-fffff write-protect
lines 1-36
```

1.2 主题 2：修改键位映射

1.2.1 内容

利用 PowerToys 可修改键位映射

1.2.2 结果

重映射键

确定取消

选择要更改的键 ("选择"), 然后配置要将其发送的键、快捷方式或文本 ("发送")。

重新映射示例 选择 A 并发送 "Ctrl+C", "A" 将是你的"选择", 而 "Ctrl+C" 将是你的"发送"命令。

选择:

发送:

选择

Win (Left)

→

发送键/快捷键

Alt (Left)

选择

🗑

选择

Alt (Left)

→

发送键/快捷键

Win (Left)

选择

🗑

+ 添加键重新映射

1.3 主题 3：守护进程

1.3.1 内容

Linux 中找出正在运行的所有守护进程
命令：systemctl status

1.3.2 结果

```
localhost.localdomain
State: degraded
Jobs: 0 queued
Failed: 1 units
Since: Wed 2024-09-11 11:53:33 CST; 4h 55min ago
CGroup: /
├─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 22
├─user.slice
│ └─user-0.slice
│   └─session-1.scope
│     ├──6201 login -- root
│     ├──7371 -bash
│     ├──7788 vi .ssh/config
│     ├──8155 systemctl status
│     └─8156 systemctl status
├─system.slice
│ ├──rsyslog.service
│ │ └─7099 /usr/sbin/rsyslogd -n
│ ├──tuned.service
│ │ └─7097 /usr/bin/python2 -Es /usr/sbin/tuned -l -P
│ ├──sshd.service
│ │ └─7096 /usr/sbin/sshd -D
│ ├──postfix.service
│ │ ├──7256 /usr/libexec/postfix/master -w
│ │ ├──7260 qmgr -l -t unix -u
│ │ └─8102 pickup -l -t unix -u
│ ├──NetworkManager.service
│ │ ├──6514 /usr/sbin/NetworkManager --no-daemon
│ │ └─6901 /sbin/dhclient -d -q -sf /usr/libexec/nm-dhcp-helper -pf /var/run/dhclient-ens
│ ├──firewalld.service
│ │ └─6216 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid
│ ├──crond.service
│ │ └─6182 /usr/sbin/crond -n
│ ├──systemd-logind.service
│ │ └─6129 /usr/lib/systemd/systemd-logind
│ └─dbus.service
```

lines 1-36

1.4 主题 4：常见命令行标志参数及模式

1.4.1 内容

一些操作的使用用法可以用
指令 `-help` 来查询

1.4.2 结果

```
$ touch --help
Usage: touch [OPTION]... FILE...
Update the access and modification times of each FILE to the current time.

A FILE argument that does not exist is created empty, unless -c or -h
is supplied.

A FILE argument string of - is handled specially and causes touch to
change the times of the file associated with standard output.

Mandatory arguments to long options are mandatory for short options too.
  -a                change only the access time
  -c, --no-create    do not create any files
  -d, --date=STRING  parse STRING and use it instead of current time
  -f                (ignored)
  -h, --no-dereference
                    affect each symbolic link instead of any referenced
                    file (useful only on systems that can change the
                    timestamps of a symlink)
  -m                change only the modification time
  -r, --reference=FILE
                    use this file's times instead of current time
  -t STAMP           use [[CC]YY]MMDDhhmm[.ss] instead of current time
                    change the specified time:
                    WORD is access, atime, or use: equivalent to -a
                    WORD is modify or mtime: equivalent to -m
  --help            display this help and exit
  --version         output version information and exit

Note that the -d and -t options accept different time-date formats.

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Report any translation bugs to <https://translationproject.org/team/>
Full documentation <https://www.gnu.org/software/coreutils/touch>
or available locally via: info '(coreutils) touch invocation'
```

1.5 主题 5: Markdown 使用: 设置标题

1.5.1 内容

使用 * 加标题名来设置

* 数目的多少代表大小标题

```
✓ # 标题一
✓ ### 标题二
✓ ##### 标题三
✓ ##### 标题四
✓ ##### 标题五
```

1.5.2 结果

- 标题一
- 标题二
- 标题三
- 标题四
- 标题五

1.6 主题 6：Markdown 使用：粗体、斜体

1.6.1 内容

粗体: **** 粗体内容 ****

斜体: ** 斜体内容 **

```
**粗体内容**  
*斜体内容*
```

1.6.2 结果

- 粗体内容
- 斜体内容

1.7 主题 7：Markdown 使用：引用块、代码块

1.7.1 内容

引用

```
> 引用
```

代码块

```
#代码块
```python
import matplotlib.pyplot as plt

labels = ['China', 'Japan', 'US', 'UK']
fracs = [70, 10, 10, 10] # 这些是百分比，但我们需要将它们转换为浮点数
fracs = [float(i) / 100 for i in fracs] # 转换为百分比形式（0-1之间）

plt.pie(x=fracs, labels=labels, autopct='%1.1f%%') # 使用 '%1.1f%%'
来格式化百分比
plt.show()
```
```

1.7.2 结果

> 引用

#代码块

```
import matplotlib.pyplot as plt

labels = ['China', 'Japan', 'US', 'UK']
fracs = [70, 10, 10, 10] # 这些是百分比，但我们需要将它们转换为浮点数
fracs = [float(i) / 100 for i in fracs] # 转换为百分比形式（0-1之间）

plt.pie(x=fracs, labels=labels, autopct='%1.1f%%') # 使用 '%1.1f%%' 来格式化百分比
plt.show()
```

1.8 主题 8: Markdown 使用: 列表

1.8.1 内容

有序列表

- ```
1. First item
2. Second item
3. Third item
```

无序列表

- ```
- First item
- Second item
- Third item
```

1.8.2 结果

有序列表:

1. First item
2. Second item
3. Third item

无序列表:

- First item
- Second item
- Third item

1.9 主题 9: Markdown 使用: 表格

1.9.1 内容

```
学号	姓名	分数
小明|男|75
小红|女|69
小华|男|66
```


1.9.2 结果

| 学号 | 姓名 | 分数 |
|----|----|----|
| 小明 | 男 | 75 |
| 小红 | 女 | 69 |
| 小华 | 男 | 66 |

1.10 主题 10：Markdown 使用：数学公式

1.10.1 内容

- 1.行内公式：使用两个”\$”符号引用公式:
- 2.行间公式：使用两对“\$\$”符号引用公式:

1.10.2 结果

```
$\sqrt{x^2}$  
$$\sqrt{x^2}$$
```

$\sqrt{x^2}$

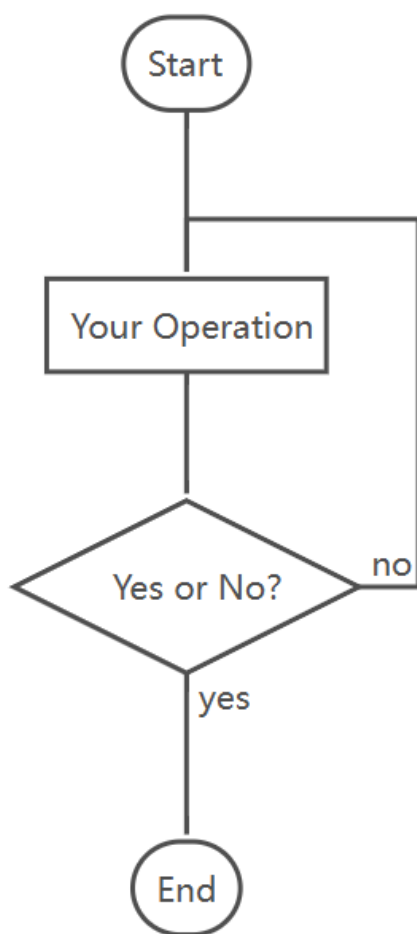
$\sqrt{x^2}$

1.11 主题 11: Markdown 使用: Mermaid 流程图

1.11.1 内容

```
```flow
st=>start: Start
op=>operation: Your Operation
cond=>condition: Yes or No?
e=>end
st->op->cond
cond(yes)->e
cond(no)->op
```

### 1.11.2 结果



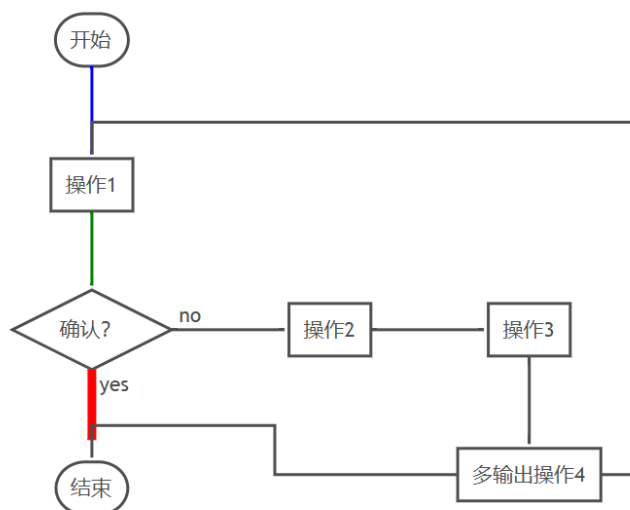
## 1.12 主题 12: Markdown 使用: Flowchart 流程图

### 1.12.1 内容

```
```mermaid
flowchart
st=>start: 开始
e=>end: 结束
op1=>operation: 操作1 | past
op2=>operation: 操作2 | current
op3=>operation: 操作3 | future
pa=>parallel: 多输出操作4 | approved
cond=>condition: 确认? | rejected

st->op1->cond
cond(true)->e
cond(no)->op2(right)->op3->pa(path1,right)->op1
pa(path2,left) ->e
st@>op1({"stroke":"Blue"})@>cond({"stroke":"Green"})@>e
({"stroke":"Red","stroke-width":6,
"arrow-end":"classic-wide-long"})
```
```

### 1.12.2 结果

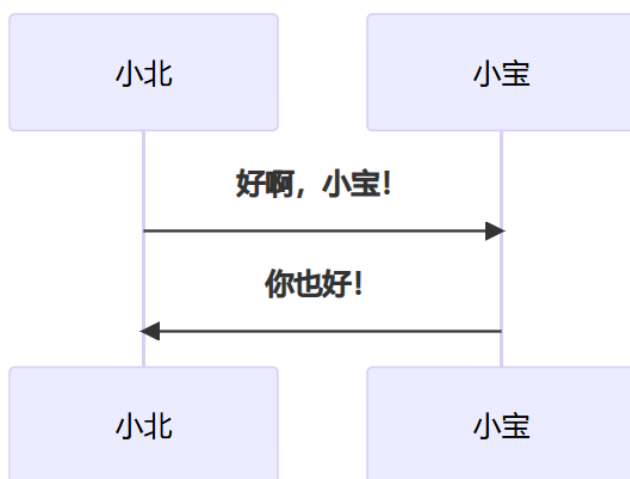


## 1.13 主题 13: Markdown 使用: 时序图

### 1.13.1 内容

```
```\nmermaid\nsequenceDiagram\n    participant 小北\n    participant 小宝\n    小北->>小宝: 好啊, 小宝!\n    小宝->>小北: 你也好!\n```\n
```

1.13.2 结果

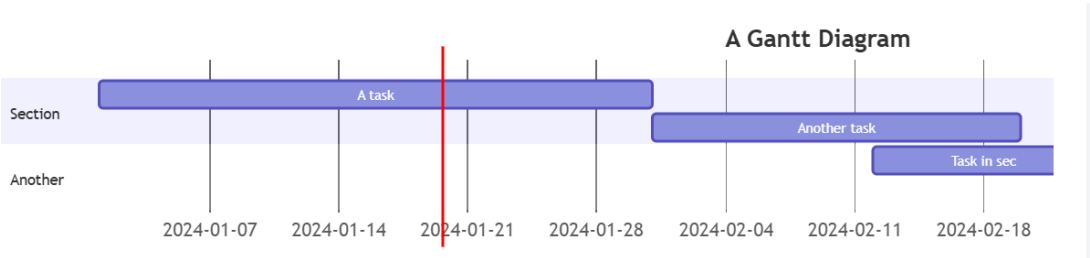


1.14 主题 14: Markdown 使用: 甘特图

1.14.1 内容

```
```mermaid
gantt
 title A Gantt Diagram
 section Section
 A task :a1, 2024-01-01, 30d
 Another task :after a1 , 20d
 section Another
 Task in sec :2024-02-12 , 12d
 another task : 24d
    ```
```

1.14.2 结果

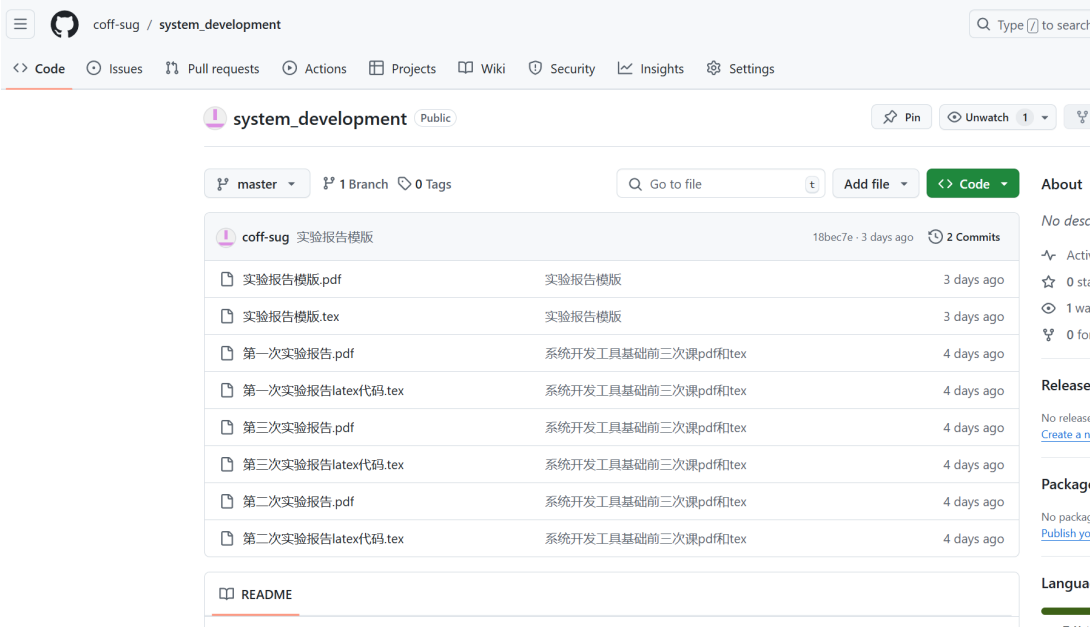


1.15 主题 15: github 的使用

1.15.1 内容

在 github 上创建仓库提交文件

1.15.2 结果



1.16 主题 16: pytorch 入门

1.16.1 内容

`torch.randperm(n)` 函数用于生成一个从 0 到 n-1 的随机排列

```
x = torch.randperm(10)
print (x)
```

1.16.2 结果

```
tensor([6, 3, 9, 0, 1, 4, 8, 7, 2, 5])
```

1.17 主题 17: pytorch 入门

1.17.1 内容

`torch.eye(3)` 生成单位矩阵

`torch.zeros((2, 3))` 生成零矩阵

`torch.rand((2, 2))` 生成随机分布矩阵

```
# 生成一个3x3的单位矩阵
I = torch.eye(3)
print(I)

# 生成一个2x3的零矩阵
Z = torch.zeros((2, 3))
print(Z)

# 生成一个2x2的均匀分布随机矩阵
R = torch.rand((2, 2))
print(R)
```

1.17.2 结果

```
tensor([[1., 0., 0.],
        [0., 1., 0.],
        [0., 0., 1.]])
tensor([[0., 0., 0.],
        [0., 0., 0.]])
tensor([[0.1375, 0.4733],
        [0.6769, 0.5968]])
```


1.18 主题 18: pytorch 入门

1.18.1 内容

数学运算

```
# 基本算术运算
a = torch.tensor([1.0, 2.0, 3.0])
b = torch.tensor([4.0, 5.0, 6.0])
c = a + b # [5., 7., 9.]
print(c)

# 幂运算和对数运算
x = torch.tensor([2.0, 4.0])
y = torch.pow(x, 2) # [4., 16.]
print (y)
z = torch.log(y) # [ln(4.), ln(16.)]
print (z)

# 近似和取整运算
w = torch.tensor([3.14, 4.56])
floor_w = w.floor() # [3., 4.]
print (floor_w)
ceil_w = w.ceil() # [4., 5.]
print (ceil_w)

# 指数和开方运算
e_power = torch.exp(torch.tensor([1.0, 2.0])) # [e, e^2]
print (e_power)
sqrt_w = w.sqrt() # [sqrt(3.14), sqrt(4.56)]
print (sqrt_w)
```

1.18.2 结果

```
tensor([5., 7., 9.])
tensor([ 4., 16.])
tensor([1.3863, 2.7726])
tensor([3., 4.])
tensor([4., 5.])
tensor([2.7183, 7.3891])
tensor([1.7720, 2.1354])
```

1.19 主题 19: pytorch 入门

1.19.1 内容

stack 为拼接函数，函数的第一个参数为需要拼接的 Tensor，第二个参数为细分到哪个维度

```
A=torch.IntTensor([[1,2,3],[4,5,6]])
B=torch.IntTensor([[7,8,9],[10,11,12]])
C1=torch.stack((A,B),dim=0) # or C1=torch.stack((A,B))
C2=torch.stack((A,B),dim=1)
C3=torch.stack((A,B),dim=2)
C4=torch.stack((A,B),dim=-1)
print(C1,C2,C3,C4)
```

1.19.2 结果

```
tensor([[[[ 1,  2,  3],
          [ 4,  5,  6]],

        [[ 7,  8,  9],
          [10, 11, 12]]], dtype=torch.int32) tensor([[[[ 1,  2,  3],
          [ 7,  8,  9]],

        [[ 4,  5,  6],
          [10, 11, 12]]], dtype=torch.int32) tensor([[[[ 1,  7],
          [ 2,  8],
          [ 3,  9]],

        [[ 4, 10],
          [ 5, 11],
          [ 6, 12]]], dtype=torch.int32) tensor([[[[ 1,  7],
          [ 2,  8],
          [ 3,  9]],

        [[ 4, 10],
          [ 5, 11],
          [ 6, 12]]], dtype=torch.int32)
```

1.20 主题 20: pytorch 入门

1.20.1 内容

使用自动微分机制配套使用 SGD 随机梯度下降来求最小值

```
import numpy as np
import torch

# f(x) = a*x**2 + b*x + c 的最小值
x = torch.tensor(0.0, requires_grad=True) # x需要被求导
a = torch.tensor(1.0)
b = torch.tensor(-2.0)
c = torch.tensor(1.0)
optimizer = torch.optim.SGD(params=[x], lr=0.01) #SGD为随机梯度下降
print(optimizer)

def f(x):
    result = a * torch.pow(x, 2) + b * x + c
    return (result)

for i in range(500):
    optimizer.zero_grad() #将模型的参数初始化为0
    y = f(x)
    y.backward() #反向传播计算梯度
    optimizer.step() #更新所有的参数

print("y=", y.data, ", ", "x=", x.data)
```

1.20.2 结果

```
SGD (
  Parameter Group 0
    dampening: 0
    differentiable: False
    foreach: None
    fused: None
    lr: 0.01
    maximize: False
    momentum: 0
    nesterov: False
    weight_decay: 0
)
y= tensor(0.) ; x= tensor(1.0000)
```

2 解题感悟

这次实验内容很多，特别是大杂烩，而 pytorch 又是深度学习的内容，之前从来没有接触过，所以进度一直很慢。结束之后就发现收获还是很大的，为我以后的学习打下了一定的基础。希望以后能有更多的机会学习这种新知识。[github 地址](#) [\[点击\]](#)