

Linux 101 进程 服务 任务*

- 进程是什么
- 作业控制
- 进程的监控与管理
- 服务是什么
- 服务的监控与管理
- 定时任务（一次性、周期性）

*以 Ubuntu 16.04 为例

走近进程 (1)

Shell 命令 *

游戏

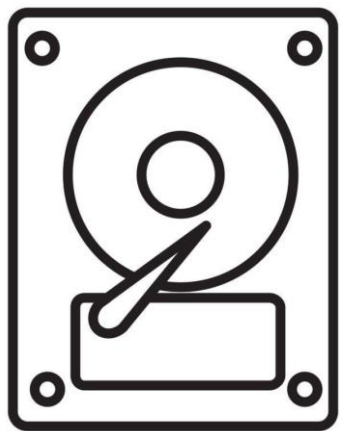
聊天软件

桌面环境

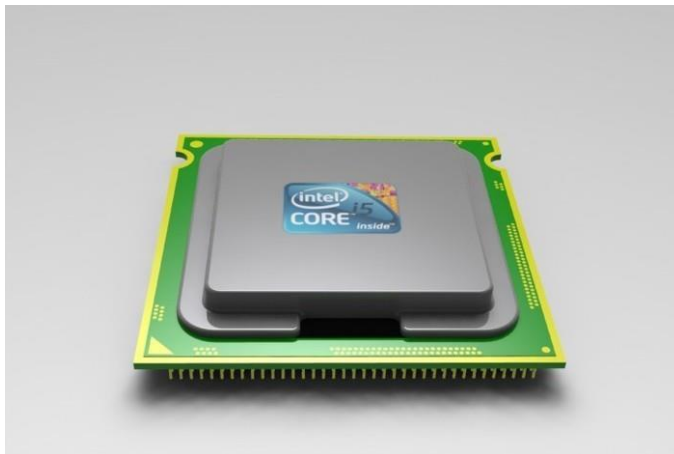
系统服务

办公套件

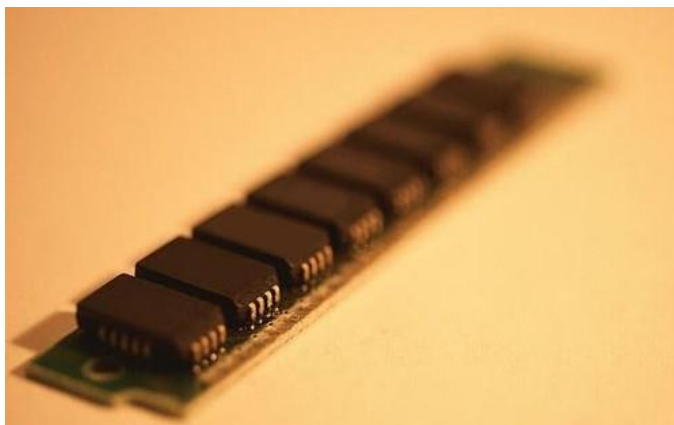
走近进程 (2)



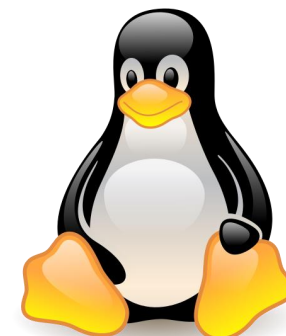
硬盘



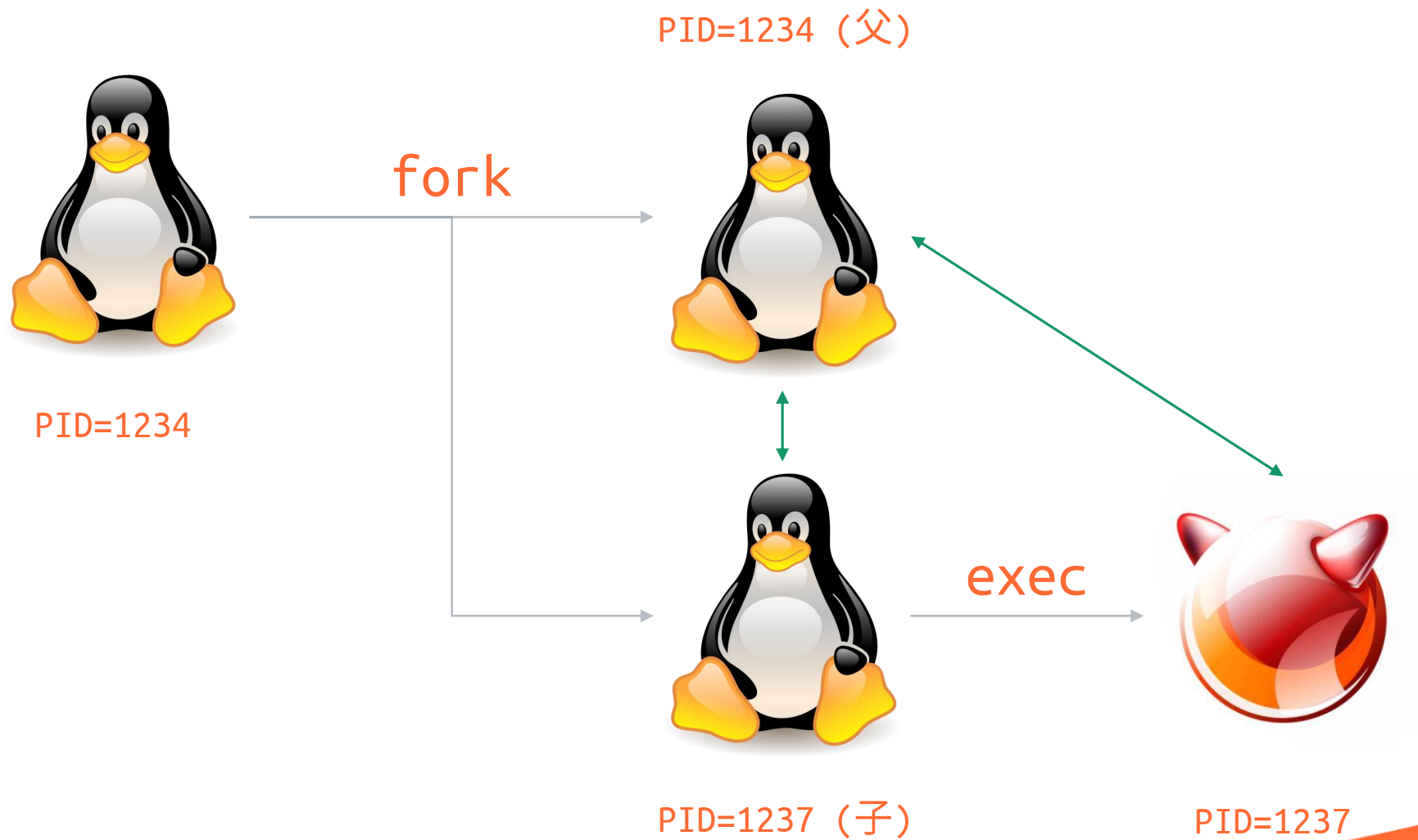
CPU



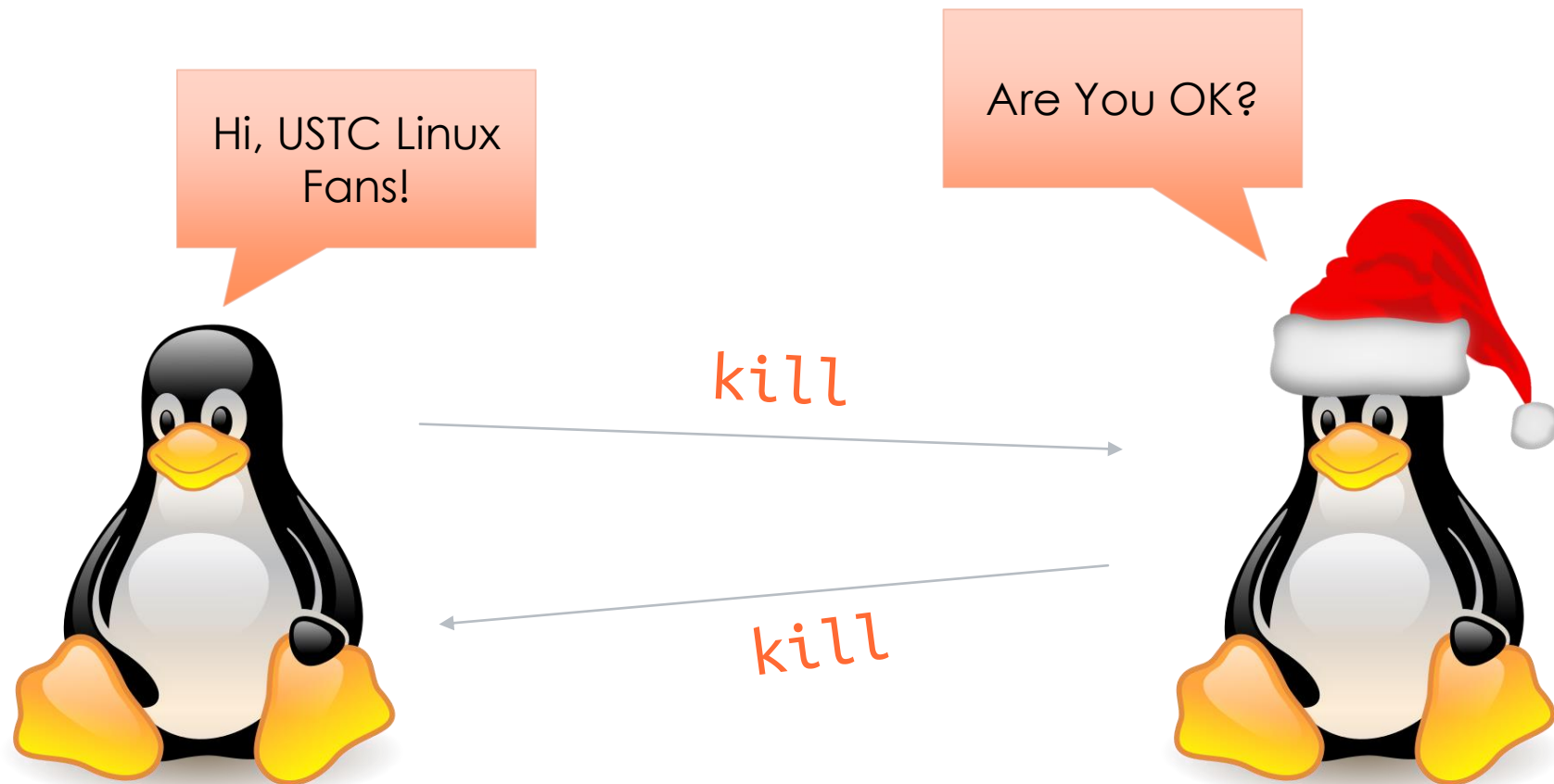
内存



走近进程 (3)



信号 (1)



信号 (2)

\$ man 7 signal

常用信号	意义	默认行为	产生手段
SIGINT (interrupt)	朋友，别干了	终止进程	Ctrl-C
SIGTERM (termiate)	请优雅地死去	终止进程	kill <PID> pkill <进程名>
SIGKILL (kill)	请立即去世	终止进程	kill -9 <PID> pkill -9 <进程名>
SIGSEGV (segment violation)	你想知道的太多了	核心转储	什么都不用做，这是程序写得太烂的缘故
SIGSTOP (stop) SIGTSTP (stop)	让某个进程变成植物人	停止进程	Ctrl-Z
SIGCONT (continue)	让植物人苏醒	继续进程	fg bg

信号 (3) 发送信号之 kill 和 pkill

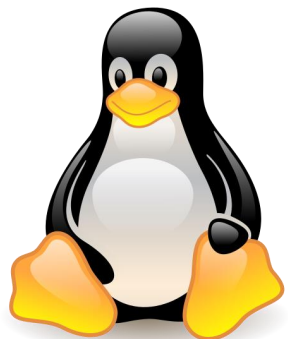
user@hostname:~\$ kill -<信号> <PID>

user@hostname:~\$ pkill -<信号> <进程名>

- 信号可以使用数字和名称两种形式表示
 - kill \$PID: 向 PID 发送 SIGTERM 信号
 - kill -9 \$PID: 向 PID 发送 SIGKILL
 - pkill -HUP rsyslogd: 向 rsyslogd 发送 SIGHUP 信号
- pkill 默认会向所有名字匹配的进程发送信号
- pgrep 找出进程的 PID 但不发送信号

作业

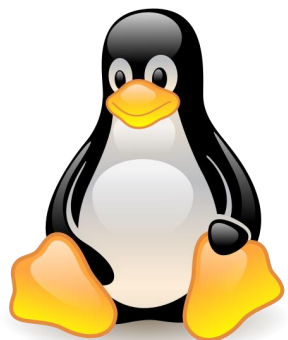
用户输入



输出

```
$ cat
```

用户输入



输出

```
$ cat | grep abc | grep xyz
```


Wait a minute...

- 连上实验室 SSH，貌似只有一个命令行界面……
- 如果这个作业很花时间，那在这期间我岂不是什么都做不了？
- 如何让多个作业同时运行？
- 如何暂停一个作业，然后恢复运行？



作业控制 (1) 前台作业与后台作业

```
user@hostname:~$ ping www.ustc.edu.cn
```

```
PING www.ustc.edu.cn (218.22.21.21) 56(84) bytes  
of data.
```

(下略)

就这么静静地看着输出，时光静好

作业编号为 1

管线中最后一个进
程的 PID 为 123

```
user@hostname:~$ ping www.ustc.edu.cn &  
[1] 123
```

表示该作业
在后台执行

```
user@hostname:~$ PING www.ustc.edu.cn  
(218.22.21.21) 56(84) bytes of data.  
(下略)
```

好了，可以继续工作了

好像没法终止？

作业控制 (2) 暂停前台作业

```
user@hostname:~$ ping www.ustc.edu.cn
```

```
PING www.ustc.edu.cn (218.22.21.21) 56(84) bytes of data.
```

```
64 bytes from 202.38.64.246: icmp_seq=1 ttl=62 time=1220 ms
```

```
^Z
```

```
[2]+  Stopped
```

```
ping www.ustc.edu.cn
```

```
user@hostname:~$
```

- 用 Ctrl-Z 停止一个前台作业。
- 暂停的作业是后台作业吗？
- 暂停的进程被终止了吗？
- 能不能用 Ctrl-Z 停止后台作业？

作业控制 (3) 列出当前作业 jobs

```
user@hostname:~$ jobs
```

```
[1]-  Stopped
```

```
[2]+  Stopped
```

```
user@hostname:~$
```

作业编号

作业状态

```
ping www.ustc.edu.cn
```

```
ping www.ustc.edu.cn
```

作业的完整命令

- 用 jobs 列出当前作业。
- + 表示 “当前作业” （如果有 Stopped，则是最近停止的作业；如果全部作业都在运行，则是最近启动的作业）。
- - 表示当前作业退出后会成为当前作业的作业。
- jobs 列表中有没有可能出现状态为 Running 的作业？ 试举一例。

作业控制 (3) 列出当前作业 jobs

参数	意义
-l	在默认输出中一并列出进程 ID。
-n	列出自上次 <u>通知</u> 以来，状态有改变的作业。
-p	只列出作业领导的进程 ID。
-r	只显示运行的作业。
-s	只显示停止的作业。

作业控制 (4) 在后台恢复作业执行 bg

```
user@hostname:~$ jobs
```

```
[1]-  Stopped
```

```
[2]+  Stopped
```

```
user@hostname:~$ bg %1
```

```
user@hostname:~$ bg
```

```
ping www.ustc.edu.cn
```

```
ping www.ustc.edu.cn
```

在后台恢复执行作业 1

在后台恢复执行当前作业
(“当前作业”可以用 %% 或 %+ 表示)

- `bg <jobspec>` 在后台恢复执行作业
- %- 表示什么作业?
- 可以使用 `bg` 将前台任务放到后台吗?

作业控制 (5) 在前台执行作业 fg

```
user@hostname:~$ jobs
```

```
[1]-  Stopped
```

```
[2]+  Stopped
```

```
user@hostname:~$ fg sleep
```

```
user@hostname:~$ fg ping
```

```
sleep 10
```

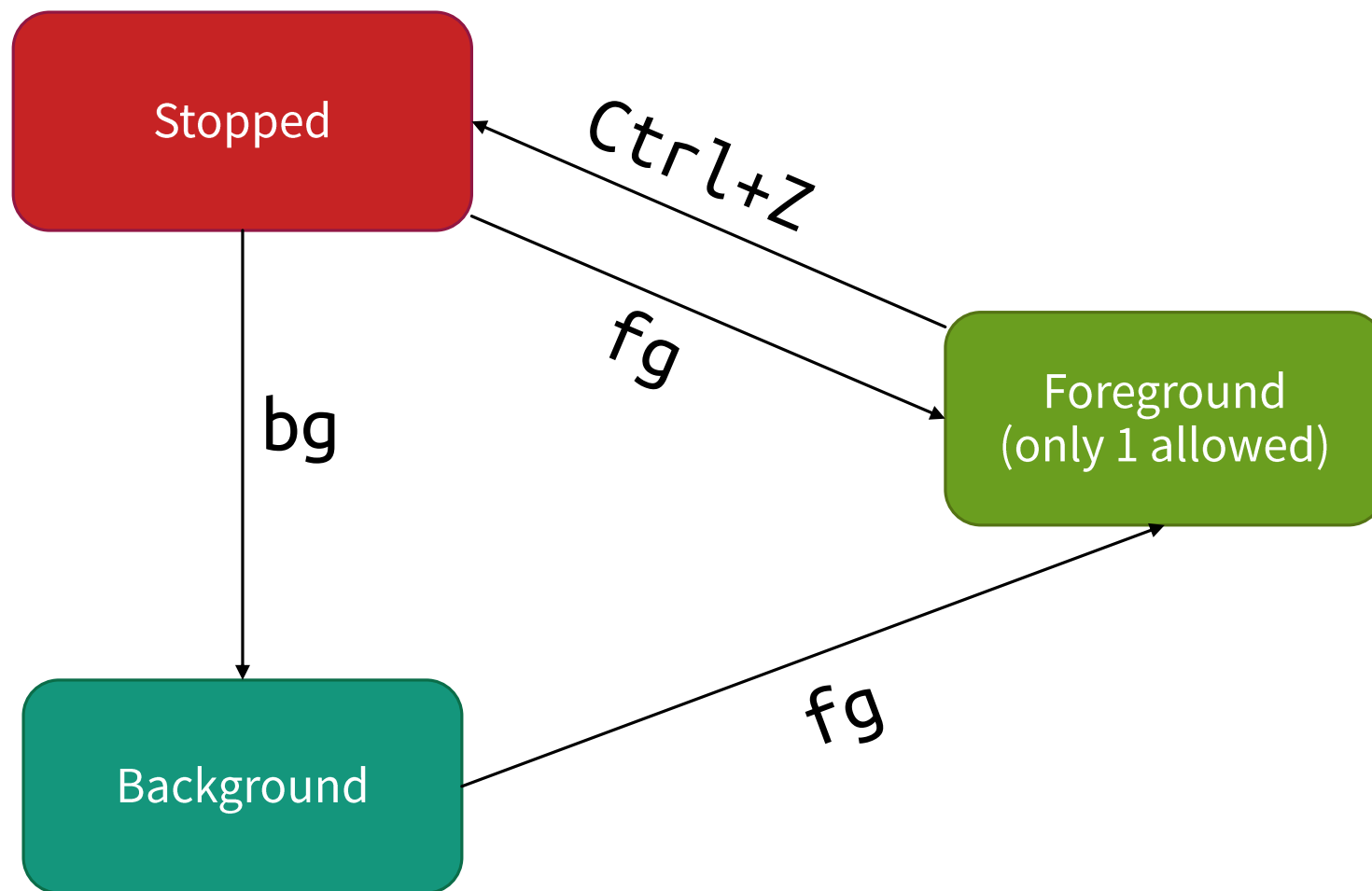
```
ping www.ustc.edu.cn
```

恢复执行 sleep 命令

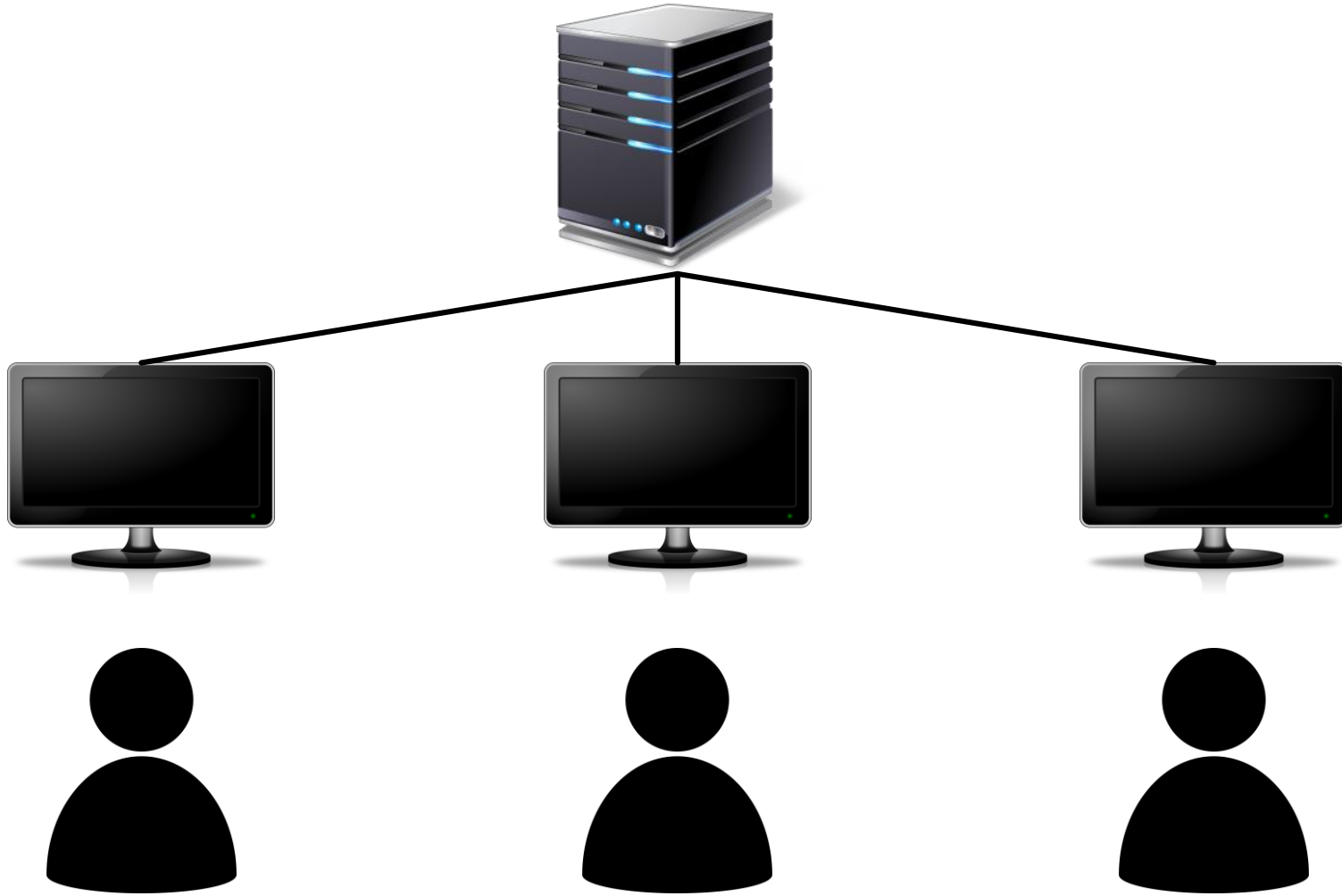
恢复执行 ping 命令

- fg <jobspec> 在前台执行作业
- bg 也可以用命令名
- fg 可以无参数，也可以用 %<编号> 和 %% 和 %+ 和 %- 作为参数
- 如果有两个命令同名的作业，这样做会发生什么？
- 可以用 fg 将后台任务放到前台吗？
- 试总结 jobspec 的语法

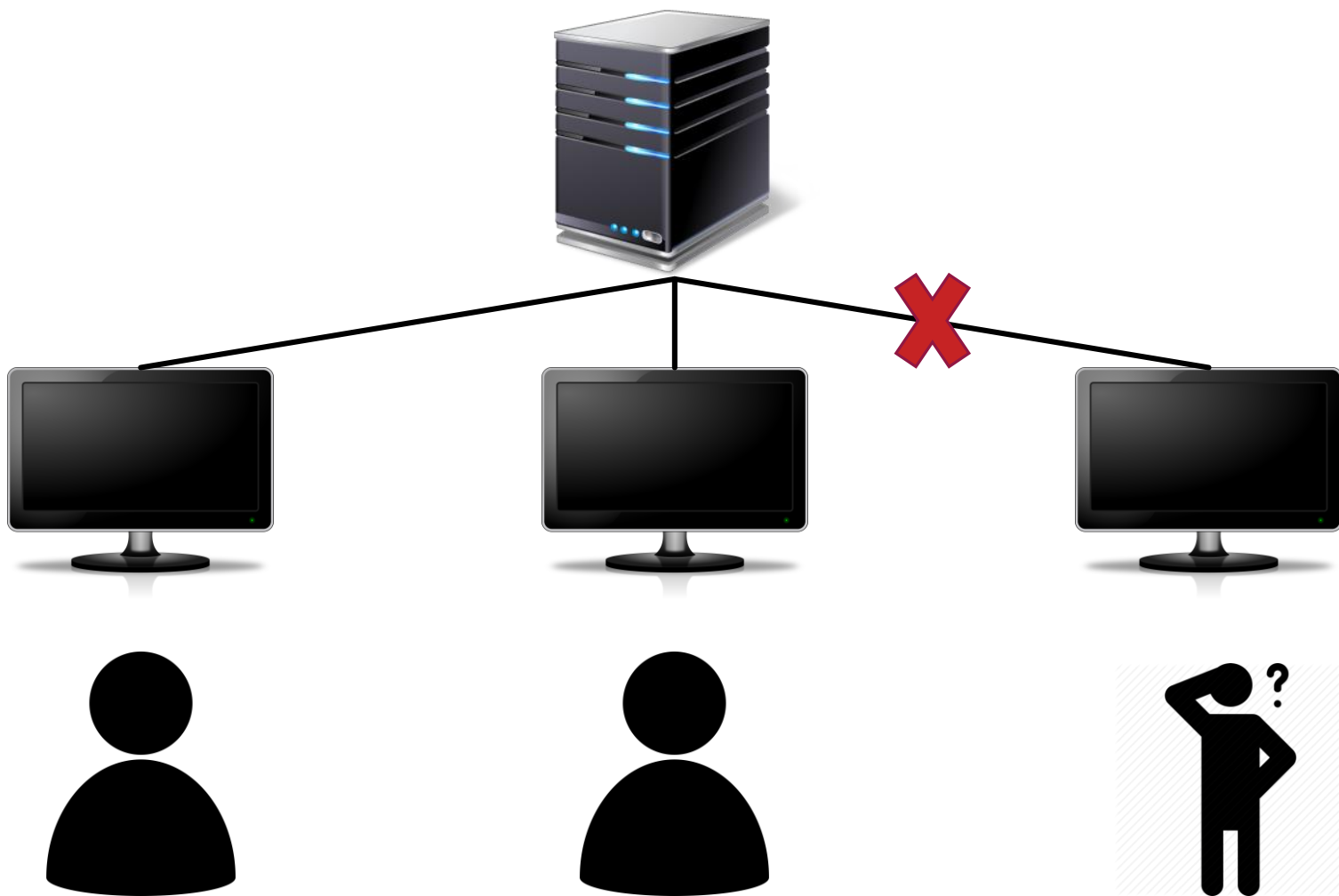
作业控制 (6) 作业状态图



一点历史 (1)



一点历史 (2)



Wait a minute...

- 如果我确实想让程序一直运行怎么做？
- 这种模型和“我通过 SSH 连接到实验室机器”似乎没有什么区别
- 那么一旦我断开 SSH 连接，程序就退出了，怎么办？



脱离控制 (2) nohup

```
user@hostname:~$ nohup ping www.ustc.edu.cn
```

```
nohup: ignoring input and appending output to 'nohup.out'  
^C
```

```
user@hostname:~$ cat nohup.out
```

```
PING www.ustc.edu.cn (202.38.64.246) 56(84) bytes of data.
```

```
64 bytes from 202.38.64.246: icmp_seq=1 ttl=62 time=3.94 ms
```

```
...(省略)...
```

```
--- www.ustc.edu.cn ping statistics ---
```

```
8 packets transmitted, 7 received, 12% packet loss, time 8871ms
```

```
rtt min/avg/max/mdev = 3.942/17.762/30.678/8.823 ms
```

- 忽略 SIGHUP 信号
- 如果 stdin 是终端 -> 重定向到一个不可读的文件
- 如果 stdout 是终端 -> 追加输出到 nohup.out

tmux (1)

```
ksqsf@verido:~$ █
```

← 普通的命令行环境

窗口号

↓

```
[0] 0:bash* "verido" 20:14 16-Apr-18
```

↑ ↑ ↑ ↑ ↑

会话号 窗口名 主机名 时间 日期

tmux (2)

普通的命令行环境 1 号

ksqsf@verido:~\$

普通的命令行环境 2 号

ksqsf@verido:~\$

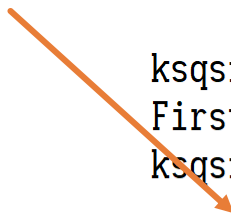
[0] 0: bash*

"verido" 20:21 16-Apr-18

在右边新建窗口 C-b %

tmux (3)

普通的命令行环境 1 号



```
ksqsf@verido:~$ echo First!  
First!  
ksqsf@verido:~$
```

```
ksqsf@verido:~$ echo Second!  
Second!  
ksqsf@verido:~$ █
```

```
[0] 0: bash* "verido" 20:25 16-Apr-18
```

普通的命令行环境 2 号

在下边新建窗口 C-b "

tmux (4)

- 进程和终端脱节
- 完全使用 tmux session / window 机制管理 shell
- 可以把一个终端上的会话传递到另一个终端 (detach / attach)
- 超高可定制性
- 更多功能，有待你的发掘

快捷键	功能
C-b ?	列出所有快捷键
C-b C-z	暂停当前 tmux
C-b &	杀死当前窗口
C-b 0~9	选择 0~9 窗口
C-b \$	重命名当前会话
C-b ,	重命名当前窗口

查询进程状态 ps (1)

```
user@hostname:~$ ps
```

PID	TTY	TIME	CMD
2	tty1	00:00:00	bash
125	tty1	00:00:00	ping
126	tty1	00:00:00	ping
137	tty1	00:00:00	ping
546	tty1	00:00:00	ps

进程
ID

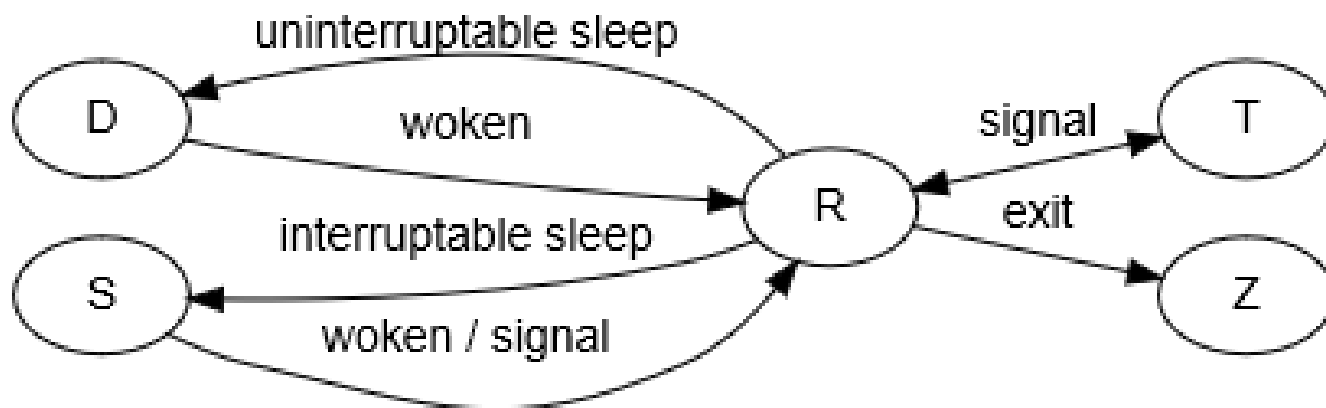
与进程关联
的终端

CPU 时间

进程命令

查询进程状态 ps (2)

状态	ps 表示	意义
Running	R	被调度运行
Interruptible	S	可中断的睡眠
Uninterruptible	D	不可中断的睡眠
Traced / Stopped	T	被追踪 / 暂停
Zombie	Z	僵死



查询进程状态 ps (3)

进程筛选

选项	将选中
-e / -A	所有进程
-a	除 session leader 和没有关联终端的进程外的所有进程
-d	除 session leader 外的所有进程
-N	除了满足选择条件外的所有进程（即反选）

查询进程状态 ps (4)

进程筛选

选项	将选中
-C <cmdlist>	可执行文件名在 cmdlist 中的进程
-p <pidlist>	PID 在 pidlist 中的进程
-s <sesslist>	会话 ID 在 sesslist 中的进程
-t <ttylist>	终端 ID 在 ttylist 中的进程
-U <userlist>	真实用户 ID 或用户名
-u <userlist>	有效用户 ID 或用户名
-G <grplist>	真实组 ID 或组名
-g <grplist>	有效组 ID 或组名

查询进程状态 ps (5)

输出格式控制

选项	输出格式
-f	全格式输出
-F	附加格式输出
-o <format>	Format = pid, s (状态), c (CPU 占用), ppid (父进程 pid), comm (命令行), ni (nice 值), pri (优先级), rss (实际内存占用) 等等，详见手册页
-O <format>	和 -o 一样，但加上了默认栏
-H	显示进程层次
--sort	[(+ -)key1,[(+ -)key2,...]]

查询进程状态 ps (6)

常见使用案例

```
user@hostname:~$ ps aux
```

```
user@hostname:~$ ps -ef
```

```
user@hostname:~$ ps -U root
```

```
user@hostname:~$ ps -U root -0 %mem --sort +%mem
```

```
user@hostname:~$ ps -C httpd -0 %mem
```

```
user@hostname:~$ ps -C uwsgi,python -0 %mem
```

```
user@hostname:~$ ps -C SomeMatrixCalc -0 %cpu
```

查看系统的内存占用 (1) free

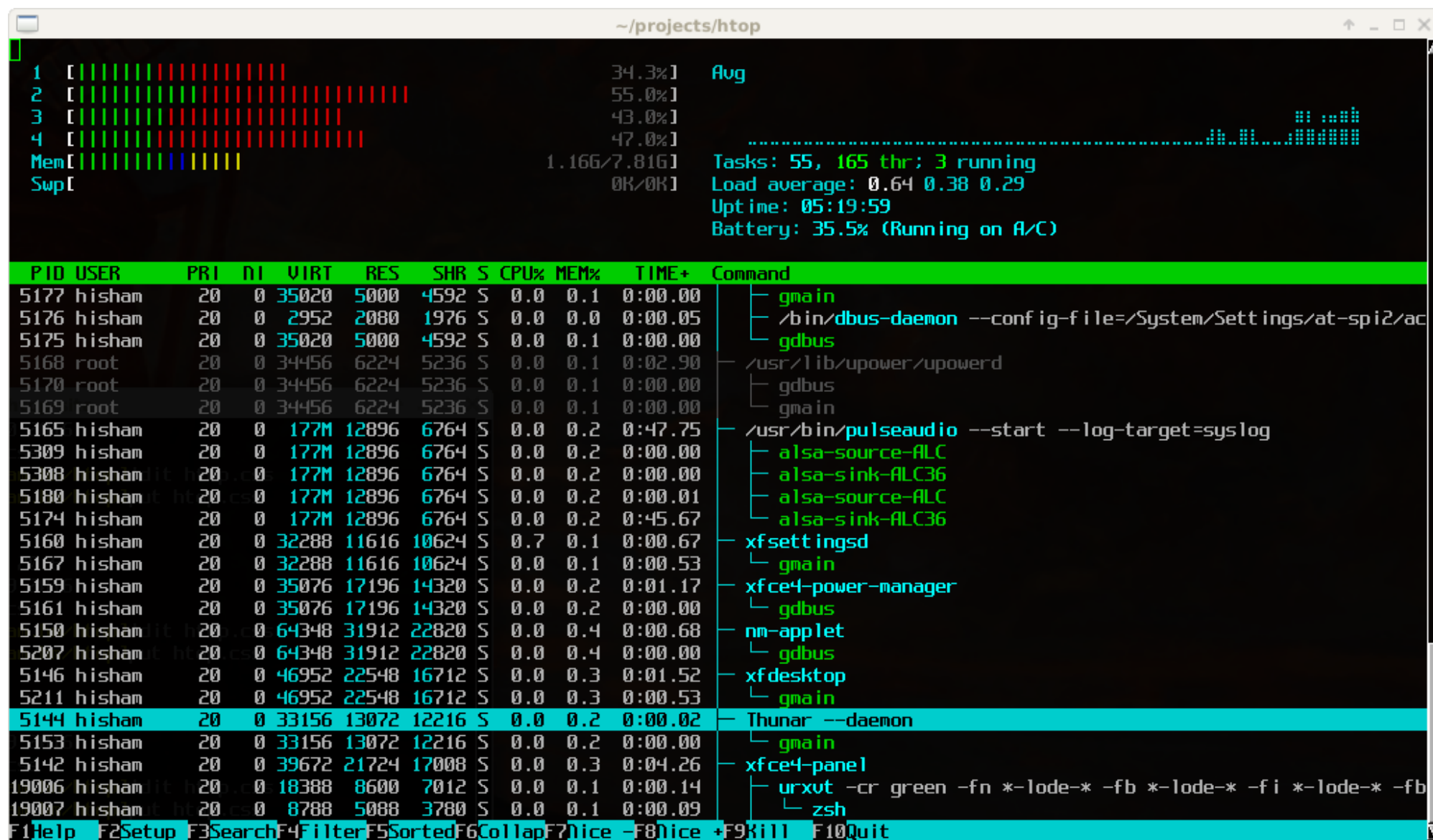
		总量		未使用	tmpfs 使用	Buffer: 内核缓冲 Cache: 页缓存和 slab	
		total	used	free	shared	buff/cache	available
内存 →	Mem:	8311900	4262036	3813388	17720	236476	3909008
交换空间 →	Swap:	25165824	45556	25120268			

已使用
(total - free - buffer - cache)

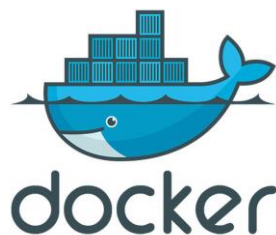
不交换的情况下新程序可用多少内存
(估计)

- 默认 KiB: -k / --kilo
- 换成其他单位: -b / --bytes, -m / --mega, -g / --giga, --tera
- 自动单位: -h / --human

查看进程的资源占用 (2) htop

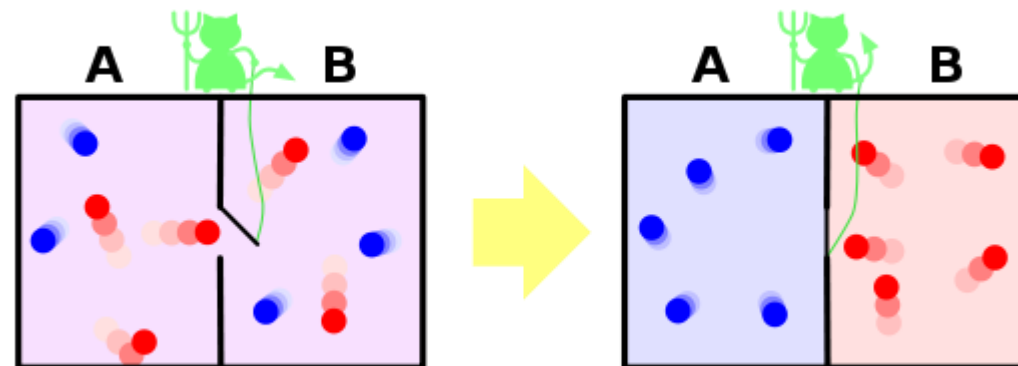


守护进程 (Daemon)



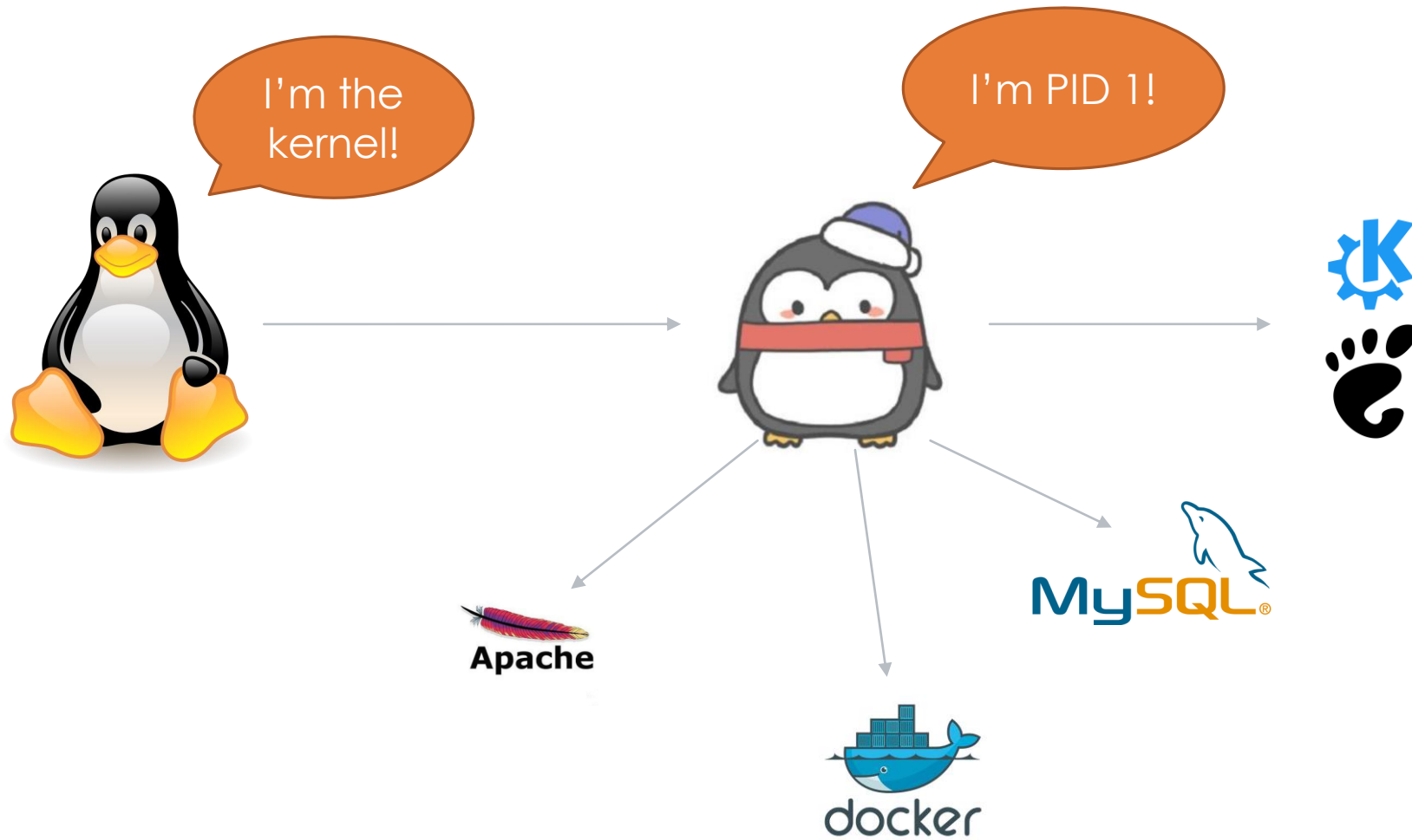
systemd logind
crond sshd acpid

Daemon
(or, system services)



Maxwell's demon

服务管理 (1) init vs systemd



服务管理 (2) 单服务常见操作

root@hostname:~# systemctl stop apache2 停止 apache2

root@hostname:~# systemctl start nginx 启动 nginx

root@hostname:~# systemctl restart ssh 重新启动 ssh

root@hostname:~# systemctl reload nginx 让 nginx 重新加载配置文件

root@hostname:~# systemctl status httpd 显示 httpd 状态

root@hostname:~# systemctl enable nginx 开机自启动 nginx

root@hostname:~# systemctl disable apache2 取消开机自启动 httpd

服务管理 (3) 列出单元、查看日志

user@hostname:~\$ systemctl list-units

root@hostname:~# journalctl [-f/--follow]
[-x/--catalog]
[-e/--pager-end]
[-u/--unit]=单元名
[-S/--since]=起始时间
[-U/--until]=截止时间
[--no-pager]

服务管理 (4) 创建自己的服务

~~编写守护进程，嚓嚓嚓.....~~

~~撰写 `.service` 文件，刷刷刷.....~~

`/etc/systemd`

系统级服务 (需放在正确目录下, 如 `user`)

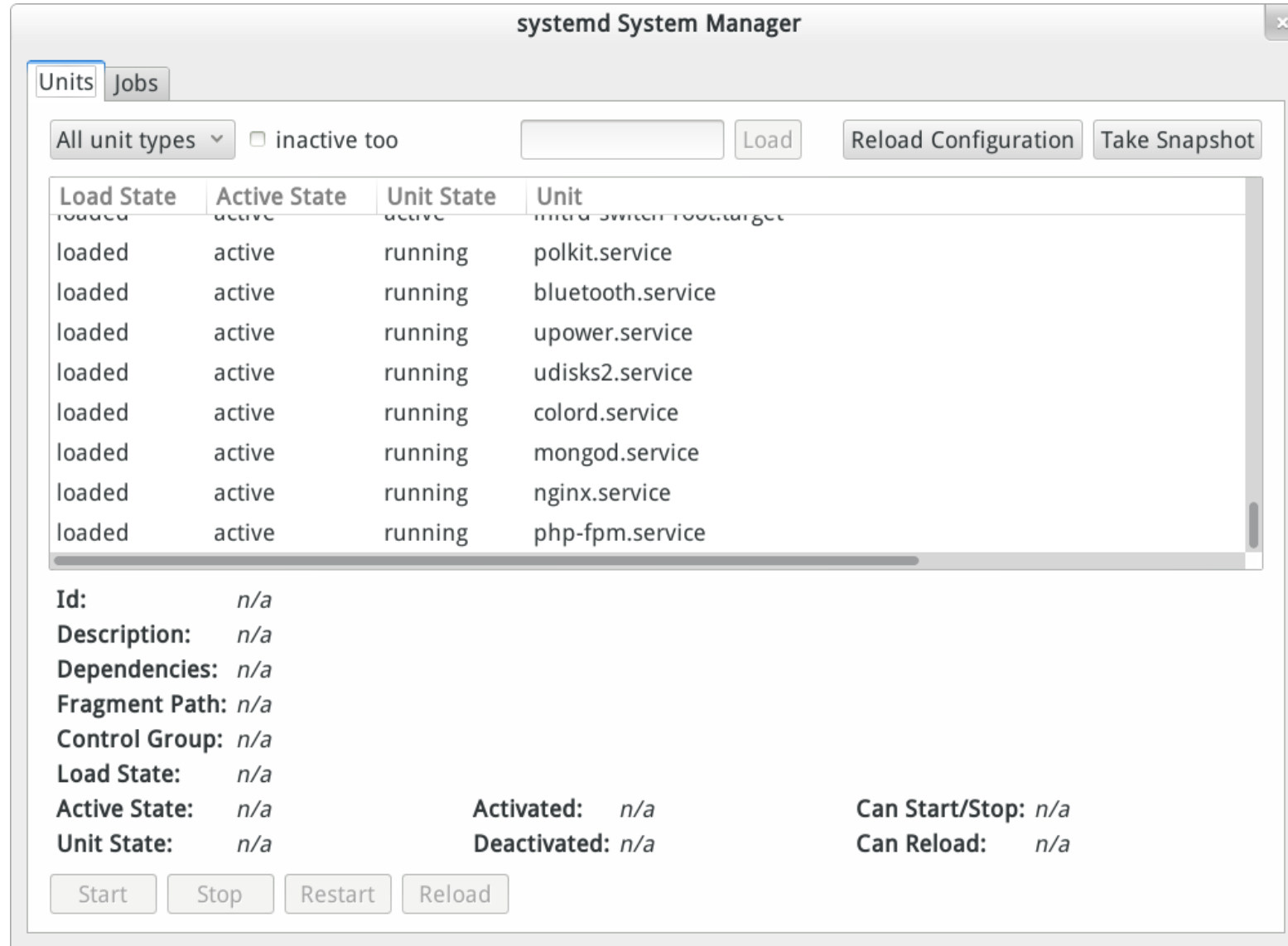
`~/.config/systemd/user`

用户级服务

```
user@hostname:~$ systemctl [--user] daemon-reload
```

```
user@hostname:~$ systemctl [--user] start my.service
```

服务管理 (5) 图形化界面 systemd-ui (systemadm)



Wait a minute...

- 服务必须由我手动启用、禁用、启动、停止……
- 如果想要监视计算的进度，能否定时周期性地在 Telegram 上通知我？
- 定时备份？
- 10 分钟后自动关机？
- 有没有一种办法，能定时执行任务？
- ~~难道要我手写 service? !~~



一次性任务 at (1)

user@hostname:~\$ at 12:30

在 12:30 执行任务

warning: commands will be executued using /bin/bash

at> echo "hello" > /home/user/message

at> <EOT>

job <N> at <DATETIME>

user@hostname:~\$ at 17:23 2018-4-13 -f script.sh

在指定时间执行脚本 script.sh

user@hostname:~\$ atq

列出当前用户的任务（若为 root，列出所有用户的任务）

user@hostname:~\$ at -c <N>

输出任务 N 脚本

user@hostname:~\$ atrm <N>

删除任务 N

一次性任务 at (2)

允许哪些用户使用 at?

<code>/etc/at.allow</code>	如果存在该文件，只有这些用户可以用 at
<code>/etc/at.deny</code>	如果存在该文件，则除此以外的用户可以用 at

(*root* 用户不受任何约束)

如果这两个文件都存在呢?

如何让所有用户使用 at?

周期性任务 (1) cron

/etc/crontab
crontab -e

系统 crontab
创建本用户的 crontab

	分钟(m)	小时(h)	Day of month (dom)	月份 (month)	Day of week (dow)	用户 (user)	命令 (command)
示例	35	15	1,3-7	JAN-FEB	*	root	rm -rf /var/MyData
取值范围	0-59	0-23	1-31	1-12, JAN-DEC	0-6, SUN-SAT	(any user)	(any shell command)

@yearly

@reboot

@monthly

@weekly

@daily

@hourly

/etc/cron.hourly

/etc/cron.daily

/etc/cron.weekly

/etc/cron.monthly

周期性任务 (2) cron

允许哪些用户使用 cron?

`/etc/cron.allow`

如果存在该文件，只有这些用户可以用 cron

`/etc/cron.deny`

如果存在该文件，则除此以外的用户可以用 cron

(*root* 用户不受任何约束)

如果这两个文件都存在呢?

如何让所有用户使用 cron?

周期性任务 (3) anacron

cron 在错过任务后，不会重新执行任务！

-> 在关机期间没必要执行的命令适合直接用 cron

-> 如果要确保命令一定被执行，可以安装 anacron

-> 安装 anacron 将禁用 /etc/cron.hourly 等目录下脚本执行

-> anacron 每小时执行一次

/etc/anacrontab

	频率	延迟	任务标识符	命令 (command)
示例	@daily	10	backup	/bin/bash /home/user/backup.sh
取值范围	@weekly, @monthly, N (每 N 天)	执行一个任务前等待分钟数	用于日志和时间戳的名称	(any shell command)

周期性任务 (4)

cron	anacron
独立使用	与 cron 配合使用
时间粒度为分钟	时间粒度为天
关机时不执行任务	下次启动执行关机期间任务
普通用户和 root 用户都可使用	只有 root 用户可以使用

... What about systemd timers?

.i ti fanmo

This is the end