

---

# Linux 资源限制

---

通过限制特定用户或程序的资源访问，也可以实现性能调优的目的。对于多用户的系统不限制资源本身可能就是一种不公平。

通过 ulimit 是限制系统资源的一种途径，ulimit 支持 hard 和 soft 限制。

Soft limits can be adjusted by the affected user.

Hard limits are set by a system administrator and cannot be changed by a regular user.

普通用户可以设置自己的软限制，但不能高于硬限制。

可以使用 ulimit -a 查看资源限制列表

```
[root@serverX ~]# ulimit -Hn 限制数 #硬限制，临时规则
```

```
[root@serverX ~]# ulimit -Sn 限制数 #软限制，临时规则
```

修改/etc/security/limits.conf 文件，实现永久规则，如

```
@test soft maxlogins 2
@test hard maxlogins 2
```

限制 test 组中的任何用户，仅可以登陆系统 2 次，大于 2 次报错（用一个用户可以登陆系统 2 次）

---

systemd 可以通过 drop-in 文件设置限制:

在/etc/systemd/system 目录下, 新建一个目录, 目录名称为 unit 文件名加.d, 如 httpd.service 这个服务

然后, 在新建的目录下创建\*.conf (文件以 conf 结尾即可)

```
[root@serverX ~]# mkdir /etc/systemd/system/httpd.service.d/
```

```
[root@serverX ~]# vim /etc/systemd/system/httpd.service.d/httpd.conf
```

```
[Service]
```

```
LimitNOFILE=32
```

```
#限制最大文件数量
```

这个文件的优先级大于/usr/lib/systemd/system 和/etc/systemd/system.

最后使用 daemon-reload, 让设置生效

```
[root@serverX ~]# systemctl daemon-reload
```

```
[root@serverX ~]# systemctl restart httpd.service
```

所有限制参数, 可以参考 **man systemd.exec !!!**

```
[root@serverX ~]# cp /usr/lib/systemd/system/sshd.service /usr/lib/systemd/system/sh.service
```

```
[root@serverX ~]# vim /usr/lib/systemd/system/sh.service
```

---

[Unit]

Description=OpenSSH server daemon

Documentation=man:sshd(8) man:sshd\_config(5)

[Service]

LimitCPU=10

Type=simple

ExecStart=/root/test.sh

ExecReload=/bin/kill -HUP \$MAINPID

KillMode=process

[Install]

WantedBy=multi-user.target

[root@serverX ~]# vim /root/test.sh

```
#!/bin/bash
```

```
while :
```

```
do
```

```
    echo ok
```

```
done
```

[root@serverX ~]# watch -n 1 "ps aux | grep test"

#仔细观察 (红色为 CPU 时间)

```
root      20068 15.8  0.1 115256 1232 ?        Ss   16:16   0:02 /bin/bash /root/test.sh
```

**注意：CPU 时间与自然时间不同！！**

---

## POSIX ulimit 参数说明

```
[root@serverX ~]# cat /etc/security/limits.conf
```

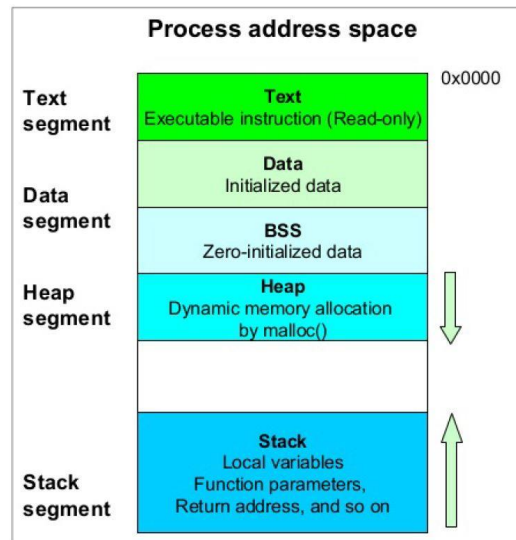
**core - limits the core file size (KB)**      限制 core 文件大小

corefile 文件是对正在运行的进程的内存镜像（主要目的是当进程崩溃后，可以通过 corefile 排错），默认值为 0（禁止 corefile）

data - max data size (KB)

限制进程得 data 内存大小

一个进程的内存空间如下图：



Text 文本：存储可执行的代码

数据：Data 初始化数据（如变量），BSS 初始化 0 数据（初始化数据为 0）

Heap（堆）通过内存的 malloc 方法，动态内存分配（一般放数据）

Stack（栈）（存储局部变量，函数等）

---

**stack - max stack size (KB)**    **限制进程使用堆栈段**

```
[root@serverX ~]# ulimit -s 16
```

```
[root@serverX ~]# ls /etc
```

```
Segmentation fault (core dumped)
```

**fsz - maximum filesize (KB)**

**限制最大文件大小**

```
[root@serverX ~]# ulimit -a
```

```
file size                    (blocks, -f) unlimited
```

**#默认为无限制**

```
[root@serverX ~]# ulimit -f 1500
```

**#通时调整软，硬限制为 1500 blocks**

```
[root@serverX ~]# ulimit -H -f
```

**#查看硬限制**

```
1500
```

```
[root@serverX ~]# ulimit -S -f
```

**#查看硬限制**

```
1500
```

---

```
[root@serverX ~]# ulimit -S -f 16000
```

#无法调整 (软限制不可以超过硬限制)

```
-bash: ulimit: file size: 无法修改 limit 值: 无效的参数
```

```
[root@serverX ~]# dd if=/dev/zero of=a bs=3M count=1
```

```
File size limit exceeded (core dumped)
```

管理员可以调整软、硬限制;

普通用户可以调整软限制 (不能超过硬限制), 仅能减小硬限制。

**memlock - max locked-in-memory address space (KB)**

最大锁定内存地址空间大小 (仅对普通用户有效, 管理员无效)

Linux 内核允许将进程暂时不需要的数据, 交换到 swap 分区, 在需要时被交换回来

但有些程序就需要将数据锁定在内存, 以提供更高的响应速度

```
[root@serverX ~]# ulimit -H -l 1000
```

(小写的 l, 1000 的单位为 KB)

```
[root@serverX ~]# ulimit -S -l 1000
```



---

**nofile** - max number of open file descriptors

账户可以打开的最大文件数量（文件描述符）

```
[root@serverX ~]# ulimit -n 3
```

```
[root@serverX ~]# cat /etc/passwd
```

```
-bash: start_pipeline: 进程组管道: Too many open files
```

**rss** - max resident set size (KB) **max memory size**, 限制进程所使用的最大物理内存大小

**as** - address space limit (KB) **virtual memory**, 限制进程所使用的虚拟内存大小

```
[root@serverX ~]# ulimit -v 8186
```

---

```
[root@serverX ~]# ls
```

```
ls: error while loading shared libraries: libc.so.6: failed to map segment from shared object: Cannot allocate memory
```

```
[root@serverX ~]# ulimit -v unlimited
```

**cpu - max CPU time (MIN)**

限制进程能使用的 CPU 时间，这个在前面 systemd 的案例中已经有演示

**nproc - max number of processes 最大进程数量 (对普通用户有效)**

```
[root@serverX ~]# vim /test.sh
```

```
#!/bin/bash
for i in `seq 20`
do
    sleep 100 &
done
```

```
[root@serverX ~]# chmod +x /test.sh
```

```
[root@serverX ~]# su - tom #切换普通用户
```

---

[tom@serverX ~]# /test.sh #执行没问题

[root@serverX ~]# ulimit -u 15 #限制进程数量为 15

[root@serverX ~]# su - tom #切换普通用户

[tom@serverX ~]# /test.sh #提示失败

/test.sh: fork: retry: 资源暂时不可用

/test.sh: fork: retry: 资源暂时不可用

priority - the priority to run user process with.(scheduling priority)

限制 nice 优先级的范围, **这个值仅对普通用户有效, 管理员无效**

在用户启动进程时, 可以通过 nice 命令调整进程的优先级

可以在-20 到 19 之间调整 (数字越大:如 19, 进程优先级越低, 数字越低:如-20, 优先级越高)

---

+++++

开启一个命令终端输入 (没有账户需要提前创建账户):

```
[root@serverX ~]# su - tom
```

```
[tom@serverX ~]# sleep 100
```

开启另一个命令终端输入:

```
[tom@serverX ~]# ps -eo pid,ni,comm | grep sleep
```

#显示进程 id, 进程优先级 (默认为 0), 进程名称

```
4238 0 sleep
```

+++++

开启一个命令终端输入:

```
[tom@serverX ~]# nice -n -10 sleep 100
```

#指定优先级启动

开启另一个命令终端输入:

```
[tom@serverX ~]# ps -eo pid,ni,comm | grep sleep
```

#显示进程 id, 进程优先级 (默认为 0), 进程名称

```
4238 -10 sleep
```

+++++

管理员设置优先级范围(软限制 19 至-1 之间, 硬限制 19 至-11 之间)

---

```
[root@serverX ~]# ulimit -H -e 30 (19-30=-11)
```

```
[root@serverX ~]# ulimit -S -e 20 (19-20=-1)
```

```
[root@serverX ~]# su -tom
```

```
[tom@serverX ~]# nice -n -2 ls /
```

nice: 无法设置优先级: 权限不够

+++++

sigpending - max number of pending signals

最大挂起的信号数量（所有用户有效），信号已经产生但未处理的信号为挂起信号

在 Linux 系统中，我们可以向程序发送信号（根据不同的信号，程序会执行不同的动作），如睡眠信号，终止信号，继续信号等

```
[root@serverX ~]# kill -l
```

#列出所有信号

msgqueue - max memory used by POSIX message queues (bytes)

---

详细队列是 IPC（进程间通信）的一种方式，如管道就可以实现进程间通信，消息队列也可以实现进程间通信。

进程间通信的方式：管道，共享内存，socket，消息队列，堆栈，信号

但不同的时，管道要求发送消息时，对方进程必须在监听管道接受消息；

而消息队列允许一个进程发送消息到队列（第一个进程可以关闭了），随后任意时间另一个进程启动，接着到队列中读取消息。

该参数设置消息队列能用的内存大小，也就是最创建多大的消息队列（默认 800KB）

**nice - max nice priority allowed to raise to values: [-20, 19]**

允许调节的 nice 值得范围，默认为-20 至 19，对普通用户有效

maxlogins - max number of logins for this user（单个用户可以登陆几次）

maxsyslogins - max number of logins on the system 系统最多允许几个人登陆，仅对普通用户有效

locks - max number of file locks the user can hold，每个用户最多可以拥有多少文件锁