tty显示当前终端位置 date > data.txt 🖯 输出到文件中 data > /dev/pts/4 ⊙ 重定向到另一个终端 标准输出和标准输入 proc/\$\$ ⊝ 是当前进程的pid 进程打开的文件 /proc/\$\$/fd ⊝/ 0 🖯 对应的为标准输入,对应金盘输入 相关知识 012文件描述符 ⊖ 1 ⊝ 对应电脑输出 ⊝ 标准输出,对应终端 2 ⊖ 标准错误 ⊖ 对应的是终端, 进程在运行的时候会打开多个文件,每个文件会有一个文件标识符,文件标识符为数 进程是通过访问这个描述符来访问这个文件 每个进程打开的文件时候限制的,1024,但是可以调整 date 1>date.txt 😊 输出重定向,把文件输出到文件中 练习 ⊖ date 2>date.txt ⊖ 因为咩有产生任何错误所以没有问题。 cat 0</etc/hosts ⊖ 輸入重定向 date 1>date.txt == date > date.txt ⊖ 覆盖 date >> date.txt ⊖ 追加 date 2>> date.txt \, 追加 date 2 > date.txt ⊖ 覆盖 错误输出 ᢒ echo "111" ⑤ 就是回应一下 重定向的文件不用事先准备他会自动创建 重定向有的时候就很恐怖的。在root用户下面。 ls /home /salkdjflkajsdflkajsdlkfj &> list.txt 🕤 &和吧错误输出和正确输出保存到相同位置 ls /home/ /aaaaa>list.txt 2>&1 案例4: 正确和错误都输入到相同位置 //混合输出 [root@tianyun ~]# Is /home/ /aaaaaaaaa &>list.txt 输出重定向 错误输出和正确输出到同一个位置( <del>紧例5</del>: 正确和错误都输入到相同位置 [root@tianyun ~]# Is /home/ /aaaaaaaaa >list.txt 2>&1 //重定向到相同的位置 垃圾桶 ⊙ /dev/null ⊙ 重定向到/dev/null 思考题 脚本中使用重定向 ⊖ 在脚本中使用重定向命令,在重定向执行的时候就可以执行了 输入重定向 ⊖ dd <dev.zero > file2.txt bs=1M count= 2 覆盖/dev/null已经没有办法进行了 普通文件和设备文件: [root@tianyun ~]# | /dev/nul /dev/sda1 /etc/hosts crw-rw-rw- 1 root root 1, 3 8月 1 06:36 /dev/null brw-rw---- 1 root disk 8, 1 8月 1 06:36 /dev/sda1 -rw-r--r--. 1 root root 158 6月 7 2013 /etc/hos 案例 🖯 管道及IO定向 字符文件是主设备号和从设备号 主设备号相同,为同一种设备类型,也可以认为是使用的同一个内核,也可以认为 kernel使用的是相同的驱动 字符文件没有大小,普通文件有大小( 从设备号 🖯 在同一类型设备中的一个序号 c ⊖ 表示字符设备 普通文件和字符文件的区别 b ⊙ 表示块设备 b ⊖ 指定块设备 c,u ⊝ 指定字符设备 mknod /dev/null c 1 3 🔾 b,c,u需要指定主设备号,从设备号 创建字符设备文件 mkond -m 666 /dev/null c 1 3 man mkond echo "111" echo "111"> file100.txt 1 ⊝ cat >>fiel2.txt ⊝ ctrl+d结束 cat >file4.txt < <EOF ⊖ 输入EOF就结束 cat >file4.txt <<-EOF ⊖ 可以识别tab键 小括号内是一个整体 (while :; do date ; sleep 2 ; done ) & ⊝ 放到后台但是在前台输出 综合案例 ⑤ (whilie :; do date ; sleep 2 ; done ) &>list.txt & ⑤ 在后台运行并且输出到文件中 后面安装源码软件时:(./configure && make && make && install) & >/dev/null (cd /boot ; ls) 不会改变当前shell cd /boot ; ls 🖯 改变当前shell 有一些命令会对当前shell产生影响,如果不希望当前shell产生影星请在subshell中执 (umask 000; touch file888) umask 000; touch file888 后面的不变 重定向输出到文件中 ⊖ files:> >> >>> 最核心的两句话,都是重定向 管道控制输出到其他进程中 ⊖ programs: | 0 ⊝ 标准输出 1 ⊝ 标准输入 2 ⊖ 标准错误 ps aux | less 🕒 把ps aux 的输出作为输入,输入到less 示例 ⊖/ yum list 列出所有包 rpm -qa 🖯 查看所有已经安装的包 排序用的 -n可以按照数值排序 sort命令 ⑤ -r倒过来排序 -k按照第几列排序 -t字段分割符 head  $\Theta$  前十个 ps aux --sort=-%cpu |head -6 |grep -v %CPU ⊝ 打印前五个cpu 统计多少种shell和每种shell有多少个 awk -F: '{print \$7}' /etc/passwd ⊝ 取出第七列 awk -F: '{print \$7}' /etc/passwd |sort 🕤 排序 案例二 ⊙ awk -F: '{print \$7}' /etc/passwd |sort | uniq ⊝ 去重 awk -F: '{print \$7}' /etc/passwd |sort | uniq -c ⊝ 统计 用管道来实现多个命令的配合 -F ⊖ 指定字段分隔符 yum -y install httpd systemctl start httpd systemctl stop firewalld 🖯 关闭防火墙 ss -an | grep :80 |awk -F":" '{print \$8}' | sort |uniq -c df | grep '/s' |awk '{print \$5}' |awk -F"%" {print %1}' 在想要保留的后面加上tee df | grep '/s' |awk '{print \$5}' | tee df.txt |awk -F"%" {print %1}' ⊝<sub>/</sub> df | grep '/s' |awk '{print \$5}' | tee df.txt |awk -F"%" {print %1}' T管道 ⊙ top -d 1 -b -n 1 | tee top.txt ⊙ 可以同时输出到文件和屏幕 read ip 🖯 輸入变量 匿名管道 🖯 同一个终端上之间互相操作,没有创建管道文件爱你 mkfifo 命名管道 <sup>⑤</sup> 可以实现不同管道之间的交互