学用 UNIX 1

UNIX初步

csh命令的基本概念

name [**-*option***] [*arguments*]	***name*** 代表命令名，**-*option*** 代表命令选项，*arguments* 代表命令参数
hostname	显示主机名
date	显示日期与时间（包括年月日，时分秒）
pwd	显示当前工作目录
set [*var*[=*value*]]	设定 **csh** 变量 *var* 并赋予其值 *value*；或显示全部 **csh** 变量
setenv [*VAR* [word]]	设定环境变量 VAR 并赋予其值 word；或显示全部环境变量
unset *var* 或 **unsetenv** *VAR*	撤消 **csh** 变量 *var* 或撤消环境变量 *VAR*
echo [**-n**] *list*	将 *list* 写至标准输出。**-n** 行尾不带换行符
history [**-r**] [*n*]	显示最近提交的 *n* 条命令。**-r** 倒排序号显示

命令参数中的通配符及命令行中常用特殊字符

*****	代表参数中任意个数的任意字符	**?**	代表参数中一个任意字符
[…**]**	代表参数中字符集内的任意一个字符。**[**A-Za-z**]**代表任意一个英文字母		
/	上下层目录或目录与文件间的间隔符；文件路径首为‘*/*’表示绝对路径		
~*username*	用户 *username* 的顶层目录		
~	本用户的顶层目录（相当**$home** 或**$HOME**）		
.	一点‘**.**’代表当前工作目录；置于文件名首为隐藏文件名		
..	两点‘**..**’代表当前工作目录的上层目录		
!!	再次提交前次提交的命令		
^*str1***^***str2***^**	将前次提交命令中的字符串 *str1* 改为字符串 *str2*，再提交		
!*str*	重提交最近曾提交过的以字符串 *str* 为首的命令		
!?*str***?**	重提交最近曾提交过的含字符串 *str* 的命令		
!*n*	重提交序号为 *n* 的命令		
!*n***:s/***str1***/***str2***/**	将序号为 *n* 的命令中的字符串 *str1* 改为字符串 *str2*，再提交		
command **<** *file*	执行命令 *command* 并将其标准输入重定向为文件 *file*		
command **>[>][!]** *file*	执行命令 *command* 并将其标准输出重定向为文件 *file*		
command **>[>]&[!]** *file*	执行命令 *command* 并将其标准输出和标准报错重定向为文件 *file*		
command1 **;** *command2*	顺次执行命令 *command1* 和 *command2*		
command1 **|** *command2*	流水执行 *command1* 和 *command2*，前者的标准输出作为后者的标准输入		
command **&**	提交命令 *command* 在后台执行并列入任务控制表		
$	冠于 **csh** 变量名或环境变量前表示其值	**#**	引导整行注释
`**…**`	内含命令，代表该命令写至标准输出的内容	**'**…**'**	内中特殊字符不具特殊意义，视同一般
"…**"**	内中空格不作分隔作用，整体视为一“字”	**:**	标号语句尾或环境变量值中的分隔符
(…**)**	内置若干字，空格有效，用于赋值；内置条件表达式用于流程控制；内置若干命令视为子进程		
****	在特殊字符左表示不作特殊字符用；在别名（由 **alias**定义）左表示不作别名；在行末表示续行		

登录

login [*username*]	在本机登录名为 *username* 的帐户
rlogin *hostname* [**-l** *username*]	登录到名为 *hostname* 远程主机上的本人或名为 *username* 的帐户
su [*username*]	在当前工作目录登录超级用户或一般用户 *username* 的帐户
passwd	修改登录密码（在机器提示下输入本人旧的和新的密码）

常用命令与应用程序

mkdir *dir …*	建立文件目录 *dir …*
cd [*dir*] 或 **chdir** [*dir*]	改变当前工作目录为 *dir*
basename *path*	截取路径 *path* 的文件名部份
dirname *path*	截取路径 *path* 的目录名部份
file *file …*	给出文件 *file …*的类型
chmod *code file …*	修改 *file …*授权码 *code*（八进制数表本人同组和其他人读写或执行权）
umask *code*	预定新文件禁止码 *code*（八进制数表本人同组和其他人读写或执行权）
ls [**-AFlst**] [*file …*]	列目录清单。**-A** 全列（包括隐藏文件），**-F** 特殊标志目录、可执行文件和符号链接，**-l** 长格式，**-s** 加注文件大小，**-t** 按时间逆序
df [*dir*]	列出各分区或目录 *dir* 所在分区的磁盘使用情况
du [*dir …*]	列出目录 *dir…*及其以下各层占用盘区的数量
cp [**-r**] *source* [*source …*] *target*	文件或目录（加**-r**）的复制
mv [**-f**] *source* [*source …*] *target*	文件或目录的移动或改名。**-f** 不顾目标将被复盖
ln [**-s**] *source* [*target*]	文件或目录的链接。加**-s** 为文件或目录符号链接，否则为文件硬链接
rm [**-f**] [**-i**][**-r** *dir …*] [*file …*]	文件或目录（加**-r**）的删除。**-f** 不顾文件的写保护权，**-i** 需确认
rmdir *dir …*	删除目录 *dir …*
touch *file …*	更新文件 *file …*的修改日期；如该文件不存在，则产生一个空文件
cat [*file…*]	顺次将文件 *file…*写至标准输出
more [*file …*]	分屏显示文件 *file…*的内容。**q** 中止浏览，**<CR>**走一行，其它键换屏
wc *file …*	列出文件 *file …*的行数、字数和字符数
cmp *file1 file2*	逐字比较文件 *file1* 与 *file2*，报告出现首个相异字符所在行号
diff *file1 file2*	逐行比较文件 *file1* 与 *file2*，分段对照列出全部相异行
find *dir* **-name** *pat*	列出目录 *dir* 及其以下各层目录中与模式 *pat* 匹配的文件名或目录名
grep [**-nil**] *pat* [*file …*]	列出 *file …*中含 *pat* 的行。**-i** 不顾大小写，**-n** 加列行号，**-l** 只列文件名
ctags *file …*	建立索引文件 **tags**，内列 *file…*中多种目标（如函数、宏等）的定义
sort [**-r**] [*file*]	对 **ASCII** 文件 *file* 进行行排序后写至标准输出。**-r** 要求倒排序
sed [**-e** *cmd*] [**-f** *cmd_file*] *file*	对文件 *file* 按命令 *cmd* 或按文件 *cmd_file* 进行处理后写至标准输出
tar (**c**|**x**|**t**)**fv** *tar-file* [*file …*]	文件 *file …*的打包(**c**)；包 *tar-file* 的恢复(**x**)；或显示包中文件清单(**t**)
gzip [**-c**] [**-d**] [**-r**] *file*	文件压缩与解压。**-c** 写至标准输出，**-d** 解压，**-r** 按目录压缩
which [*alias*] [*command*]	给出别名 *alias* 的定义或命令 *command* 的相关文件名
who [**am i**] 或 **whoami**	那些用户在上机。加 **am i** 仅列本用户情况。**whoami** 给出本用户名
w	机上用户在做什么
ps [**-au**]	列出本用户或所有用户（加**-a**）的所有进程。**-u** 加列用户名
<CTRL>C	中断当前执行的命令
<CTRL>Z 或 **stop %***jid …*	挂起当前执行的命令列入任务控制表或挂起后台执行的任务 *jid…*
jobs [**-l**]	列出任务控制表中现存任务（带**+**号者称当前任务）。**-l** 加注进程号
bg [**%***jid …*]	后台执行先前被挂起的任务 *jid …*（缺省为**%+**，即当前任务号）
fg [**%***jid …*]	前台执行任务控制表中的任务 *jid …*（缺省为**%+**，即当前任务号）
kill [**-9**] [*pid …*] [**%***jid …*]	封杀进程 *pid …*或任务控制表中任务 *jid …*，带**-9** 为强行
xterm [**-sb**] [**-font 9X75**] **&**	产生 **xterm** 类型仿真字符终端。**-sb** 带滚动条；**-font** *9X75* 为一种字体
clear	本仿真终端“清屏”
exit [(*expr*)]	退出 **csh** 进程或关闭窗口，*expr* 的值赋予父进程中的状态变量 **status**
time [*command*]	显示执行命令 *command* 的时间化费

csh 系统变量与环境变量

shell	SHELL	shell 程序的绝对路径，如/usr/bin/csh 或/bin/csh
user	USER	本用户名
home	HOME	本用户顶层目录
cwd	PWD	当前工作目录
term	TERM	终端类型
path	PATH	命令搜索路径，PATH 由系统根据用户所设 path 转换而得
	DISPLAY	显示设备名，由该设备所在主机名或 IP 地址加:0 构成

用户环境设置文件及其执行

▼ ~/.login　登录初始化文件，主要用于终端特性设置。本机登录或远程登录时自动执行一次

stty erase	'^H'	设定键盘上的<Backspace>键用于删去刚键入的一字符
stty werase	'^?'	设定键盘上的<Delete>键用于删去刚键入的一字串
stty kill	'^U'	设定键盘上的<CTRL>U 键用于删去刚键入的整行

```
if (`tty` == "/dev/console") then      如终端为/dev/console，意味着是从本地登录，则
  echo `hostname`:0 >! ~/.DISP           取主机名加":0"存入文件~/.DISP
  xhost +                                本机终端可作为任何远程机的显示设备
else                                   否则，意味着是远程登录，则
  if ( -f ~/.DISP) setenv DISPLAY `cat ~/.DISP`
                                         如文件~/.DISP 存在，将其所存之值设为环境变量 DISPLAY
endif                                  条件分支结束
```

▼ ~/.dtprofile　Common Desktop 环境设置。登录时自动执行一次

```
if [ $DISPLAY = ":0" ]; then           如变量 DISPLAY 之值为“ :0 ”，意味着是从本地登录，则
   DISPLAY = `hostname`:0                取主机名加“ :0 ”设为变量 DISPLAY
fi
echo $DISPLAY > ~/.DISP                将变量 DISPLAY 之值存入文件~/.DISP
xhost +                                本机终端可作为任何远程机的显示设备
```

▼ ~/.rhosts　远程登录密码免检授权文件。远程登录时系统查对

+	本用户从任何主机远程登录本帐户免检密码
hostname username	他用户 *username* 从 *hostname* 主机远程登录本帐户免检密码

▼ ~/.cshrc　初始化文件，用于设置 csh 变量、环境变量、别名等。新进程均按此自动初始化

set path = (/bin /usr/bin ~/bin)	设置三个路径为命令搜索路径
set cdpath = (.. ~/)	设置两个路径为 cd 命令中新工作目录的搜索路径
set history *n*	设置保存最近执行的命令条目数为 *n*
set prompt = "`hostname`-\!>"	设置命令提示符由主机名和命令顺序号组成
set filec	设置 csh 变量 filec，有此变量，当输入命令参数的头几个字符加<ESC>自动续全唯一相配的文件名
set echo	设置 csh 变量 echo，有此变量，重显输入命令行，其中通配符或变量均替换成相应值
setenv MANPATH /usr/man:/usr/openwin/man	设置两个联机求助的搜索路径
umask *022*	预定新建文件不给同组和其他用户写权利
source ~/.alias	读文件~/.alias，使该文件中定义的一批别名生效
source *file*	读环境设置文件 *file*，使该文件中的设置生效
rehash	改设搜索路径或增加命令文件后，重建搜索命令的 Hash 表

别名 alias 的定义与使用

alias [*alias defn*]	定义别名，*alias* 为 *defn* 的别名；或显示全部别名定义
which *alias*	显示别名 *alias* 的定义
unalias *alias*	撤消已定义的别名 *alias*
alias [*args*]	别名 *alias* 被视作一条命令。别名也可带“参数”*args*，所有“参数”或续于定义式尾或整个地替换定义式中的**\!***

别名定义举例

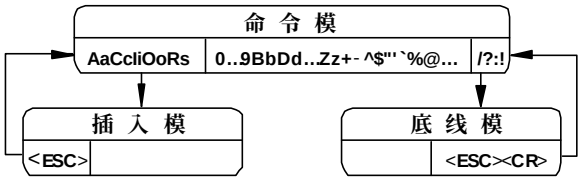
alias a alias	**a** 为命令 **alias** 的别名，用法：**a** [*alias defn*]
a u unalias	**u** 为命令 **unalias** 的别名，用法：**u** *alias*
a md mkdir	仿 **DOS**，**md** 为 **mkdir** 的别名，用法：**md** *dir …*
a +x 'chmod +x'	**+x** *file …*，改文件 *file …* 为所有用户可执行
a +w 'chmod +w'	**+w** *file …*，改文件 *file …* 为本人可写
a -w 'chmod -w'	**-w** *file …*，改文件 *file …* 为本人不可写
a rm 'rm -i'	重新定义 **rm** 为需逐一认可地删除文件。**\rm** 则保持 **rm** 的原意
a fnd 'find . -name'	**fnd** *pat*，找当前目录及其以下与模式 *pat* 相匹配的文件名或目录名
a mkae make	由于 **make** 易错输入成 **mkae**，为此，就让 **mkae** 等同 **make**
a l "\ls -F"	**l** [*file …*]，列文件目录，对其中特殊文件加标识
a l. "\ls -dF .[A-Za-z]*"	**l.**，只列隐藏文件名
a lt "\ls -Flt \!* | more"	**lt** [*file …*]，分屏以长格式并按时间倒序列文件目录
a pv 'echo \!* = $\!*'	**pv** *name*，显示名为 *name* 的 **csh** 变量或逻辑变量的定义
a ev 'env | grep \!*| more'	**ev** *str*，分屏显示逻辑变量的名或值中含 *str* 者
a cd 'set pwd = $cwd; chdir \!*; echo $cwd'	重定义 **cd** 再保存当前目录再改变并显示新工作目录，用法：**cd** [*dir*]
a , 'set back = $pwd; cd $back; unset back'	逗号‘ , ’，**cd** 到先前的工作目录
a .. 'cd ..'	两点‘ .. ’，**cd** 到当前工作目录的上层目录
a . 'echo $cwd'	一点‘ . ’，显示当前工作目录
a delete '\rm -rf \!* &'	**delete** *dir …*，在后台删除名为 *dir …* 的整个目录树
a xw"xterm -font Screen-Bold14 -geometry 80X24 -sb -sl 1000 -bd Red -bg Black -fg Yellow -ms #ffffff -cr #ffffff &"	**xw**，新开一个 **xterm** 类型窗口，指定黑底黄字带滚动条等属性
a gv '(set fl = `grep -l \!\*`; if ("$fl" != "") vi $fl)'	**gv** *pat file …*，从 *file …* 中选出含模式 *pat* 者，用 **vi** 逐个进行编辑
a vt vi -t	**vt** *tag*，根据索引文件 **tags**，用 **vi** 编辑含 *tag* 定义的文件
a rmcore '(set fl = `find $HOME -name core`; if ("$fl" != "") echo rm $fl; if("$fl" != "") \rm $fl)&'	**rmcore**，后台删除本人帐户中任何目录中的 **core** 的文件
a disp 'setenv DISPLAY \!*\:0; echo $DISPLAY >! ~/.DISP'	**disp** *addr*，设环境变量 **DISPLAY** 值为 *addr*:0，并存该值于~/.DISP
a kj 'kill %'	**kj**，封杀任务控制表中的当前任务

联机查询

man *name*	显示对 *name* 的解释，*name* 可为命令、应用程序或 C 的库函数
man intro	显示操作系统中所有命令和应用程序的清单及简介

学用 UNIX 2 熟用 vi

基本模式



进入

vi *file …*	编辑一至数个文件
vi -t *tag*	依据索引文件 **tags**，编辑含 *tag* 定义的文件，并定位于该定义的首行
vi +/*string file*	打开文件 *file*，并将光标定位于字符串 *string* 所在行
vi -R *file …*	阅读一至数个文件(仅阅读)
vi -r *file*	恢复曾被意外中断的文件编辑

光标定位

h	左移	**0**	至首列，本行第一列	**<CR>**或**+**	至下行首
j	下移	**^**	至行首，本行首个非空白列	**-**	至上行首
k	上移	**$**	至行尾，本行末列	**G**	至末行首
l 或**<SP>**	右移	*500***|**	至本行第 *500* 列	*500***G**	至第 *500* 行之首

b	至字首	**B**	至串首	**H**	至屏顶
e	至字尾	**E**	至串尾	**M**	至屏中
w	至下字首	**W**	至下串首	**L**	至屏底

%	至(、)或{、}之配偶

[[	上至位于首列的{或首行	**]]**	下至位于首列的{或末行
{	上至连续非空行前的空行	**}**	下至连续非空行后的空行
'（两单引号）	至光标先前所在行首	**`**（两反引号）	至光标先前所在位置
m*m*	标记光标位置：光标现处行称为“ *m* 行 ”；其所处位置称为“ *m* 位置 ”		
'*m*	至“ *m* 行 ”之行首	**`***m*	至“ *m* 位置 ”
f*x*	右至字符 *x*	**F***x*	左至字符 *x*
t*x*	右靠字符 *x*	**T***x*	左靠字符 *x*
;	原方向继续执行 **f**、**F**、**t**、**T**	**,**	反方向继续执行 **f**、**F**、**t**、**T**

搜索

/*pat***<CR>**	右下方向搜索字串	**?***pat***<CR>**	左上方向搜索字串
n	原方向继续搜索	**N**	反方向继续搜索

搜索为模式搜索，*pat* 代表匹配表达式（见“ **模式匹配** ”栏），也可为确定的字符串 *str*

小改

>>	整行右移一平移量	**<<**	整行左移一平移量
r*x*	改本字符为 *x*	**~**	英文字母改变大小写
J	合并下一行		

剪贴

x	删本字符	**X**	删左字符
dd	删一整行	**D**	删至行尾，相当 **d$**
yy	存一整行	**Y**	存一整行
p	贴于右下	**P**	贴于左上

剪贴命令涉及两类缓冲区：有名缓冲区"*x*（以任意一个英文字母 *x* 为名）；倒序缓冲区"#（#代表倒数序号）。前者可跨文件存取，后者不能。在删、存、贴命令前可指定有名缓冲区（如"*x***dd**，"*x***p**）；在贴命令前可指定倒序缓冲区（如"#**P**）。剪贴时未指明缓冲区，默认为倒数 *1* 号倒序缓冲区。

插入模	▼从命令模进入插入模			
	i	插于左	I	插于行首左，相当^i
	a	插于右	A	插于行尾右，相当$a
	o	下开行	O	上开行
	cc	先删去整行，相当^c$	C	先删至行尾，相当 c$
	s	先删去一字符	R	先逐字替换
	▼在插入模中			
	<CR>	换行	<BS>	删除行内新插入的字符
	<CTRL>W	删去行内新插入的一字串	<CTRL>U	全删行内新插入内容
	<CTRL>T	行首插入一平移量	<CTRL>D	行首删去一平移量
	<CTRL>V	插入不可打印字符	<ESC>	退出插入模
屏幕滚动	<CTRL>E	上滚动一行	<CTRL>Y	下滚动一行
	<CTRL>D	前滚动半幕	<CTRL>U	回滚动半幕
	<CTRL>F	前滚动一幕	<CTRL>B	回滚动一幕
	z<CR>	上滚动，本行升至屏顶	z-	下滚动，本行降至屏底
	z.	滚动，本行调至屏中央		
其它	.	重复上次修改命令	u	取消前次变化
	<CTRL>G	显示文件名、当前行号	U	整行复原
	<CTRL>L	刷新屏幕	ZZ	存盘并退出 vi
	@*x*	执行"*x* 中的命令	@@	执行先前执行的@*x*
	<CTRL>Z	中断 vi，转入 shell 状态，执行 shell 命令；用 fg 恢复 vi		
	od'*m*<ESC>"*x*dd	命令模下将一条命令 d'*m* 存入缓冲区"*x*，以备@*x* 引用		
命令扩展	▼大部分命令前可加数字，以表示该命令所含动作的重复次数。如 *3*h，*12*dd，*5*J，*4*x，*5*X，*14*s，*7*cc，*4*>>，*3*+，*2*e，*4*-，*5*w，*5*f<SP>…等等。			
	*12*dd 连删 *12* 行	*6*s 删 *6* 个字符后插入		*5*t<SP> 靠至右边第 5 个空格
	▼dd，yy，cc，>>和<<等双字符命令的第二个字符可用光标定位命令替换，以限定该命令作用范围。如 dG，db，d*4*+，d`*m*，"*p*y'*m*，d*3*t<SP>，ce，c^，c*3*-，c%，cf*x*，>'*m*，<L，>} …等等。			
	y'*m* 存本行至“ *m* 行”内容	d`*m* 删至“ *m* 位置”		cf*x* 删至右边字符 *x* 再插入
底线模	:	引导行命令，在底线输入一条完整的行命令		
	/ 或 ?	引导模式搜索，在底线输入匹配式 *pat* 或字符串 *str*		
	!	跟一个光标定位指令引导执行 shell 命令，在底线输入一条 shell 命令		
	<CR> 或<ESC>	执行上述命令，然后返回命令模		
模式匹配	▼匹配模式中的特殊字符			
	^	首列左标志	$	尾列右标志
	\<	串首左标志	\>	串尾右标志
	[*abc*]	字符集：或 *a* 或 *b* 或 *c*	[^*abc*]	字符补集：非 *a* 非 *b* 非 *c*
	.	单个任意字符(\n 除外)	\	去消右字符的特殊意义
	*	与其左字符一起代表该字符 *0* 至任意多个组成的字串		
	&	在行命令的替换命令中代表与被选择字串相同的字串		
	▼模式匹配（匹配表达式 *pat*）示例			
	\.	点	*	星号
	\/*.**\/	整条注释	\[.*\]	任意数组下标
	..*	非空整行	^$	空行
	[*A*-*Z*][*A*-*Z*]*	*1* 个至多个不间断的大写字母		
	\<[*A*-*Z*][*A*-*Z*]*\>	由 *1* 个至多个大写字母组成的独立字符串		
	[+-]*[*0*-*9*][*0*-*9]*	正负整数		

指定行	**.** 本行，即光标所在行	*3* 第 *3* 行	**.+***5* 本行下数第 *5* 行
	$ 末行	**$-** *6* 末行上数第 *6* 行	*3,12* 第 3 行至第 *12* 行
	/*pat***/** 本行下首个含 *pat* 的行	**?***pat***?** 本行上首个含 *pat* 的行	**/***s***/,/***t***/** 本行下的**/***s***/**行至**/***t***/**行
	g/*pat***/** 所有含 *pat* 的行	**v/***pat***/** 所有不含 *pat* 的行	**%** 全文范围
	:1 至首行（等效于 **1G**）	**:$** 至末行（等效于 **G**）	
字符串替换	**:s/***pat***/***str***/**	本行内首个 *pat* 替换为 *str*	
	:s/*pat***/***str***/g**	本行内所有 *pat* 替换为 *str*	
	:s/*pat***/***s1***&***s2***/g**	本行内所有 *pat* 的前后分别添加 *s1* 和 *s2*	
	:?*pat***?s/***str***/g**	本行前首个含 *pat* 行内的所有 *pat* 替换为 *str*	
	:%s/*pat***/***str***/gc**	全文中所有 *pat* 替换为 *str*，但需逐一确认	
	:g/*pat***/s/***pat1***/***str***/g**	全文中所有含 *pat* 行内的所有 *pat1* 替换为 *str*	
	:%s/^*pat***//**	删去全文中所有行首的 *pat*	
	:.,.+*5***s/^/***str***/**	本行至其下第 *5* 行，行首均前插 *str*	
	:.,$ s/*pat***$/&/***str***/**	本行至末行，行尾的 *pat* 均后续 *str*	
行编辑	**:/***pat***/+***1***,$ d**	从本行后首个含 *pat* 行的下一行起删至末行	
	:'*m***,$ t** *0*	“ *m* 行 ”至末行的内容复制于 *0* 行下，即文件首	
	:*1***,'***m* **m $**	第 *1* 行至“ *m* 行 ”的内容移至末行下，即文件尾	
	:*1***,***10* **l**	在屏幕下方列出 1 至 10 行（包括不可打印字符）；未指明行，则列当前行	
	:g/*pat***/ -** *1* **p**	在屏幕下方印出全文中所有含 *pat* 行的前一行	
	:*0* **r** *file*	读入文件 *file*，插于 *0* 行下，即文件首；未指明行，则插于当前行下	
	:'*m***,$ w** *file*	将“ *m* 行 ”至末行内容存入文件 *file*	
文件切换	**:f** [*file*]	显示正在编辑的文件名；或将其改名为 *file*	
	:args	显示进入 **vi** 时指定编辑的所有文件名	
	:n	编辑下一个文件，当编辑多个文件时用	
	:rew	返回编辑第一个文件，当编辑多个文件时用	
	:e *file* 或**:vi** *file*	编辑名为 *file* 另一个文件	
	:e# 或 **<CTRL>^**	编辑先前编辑的文件	
	:e!	重新从磁盘读入当前编辑的文件	
	:tag *tag*	依据文件 **tags**，可跨文件、跨目录定位于目标 *tag* 的原定义行	
	<CTRL>]*tag*	依据文件 **tags**，可跨文件、跨目录定位于光标后随目标 *tag* 的原定义行	
	<CTRL>T	返回先前使用**:tag** 或**<CTRL>]**命令时编辑现场	
	:w [*file*]	存盘；也可另指定存入文件名 *file*	
	:wq [*file*]	存盘并退出 **vi**；也可另存入指定文件 *file*	
	:q 或 **:q!**	退出或强行退出 **vi**，不存盘	
sh命令	**:!pwd**	执行 **shell** 命令 **pwd**，显示当前目录；击**<CR>**继续编辑	
	:r !hostname	执行 **shell** 命令 **hostname**，将其输出（主机名）插于当前行下	
	:. !\ls *.c	执行 **shell** 命令 **ls *.c**；将该命令的输出替换文本的当前行	
	!'*m***sort**	执行 **sort**，对“ *m* 行 ”至本行文本按字母顺序重新排序	
	:sh	从 **vi** 转入 **shell** 状态，执行 **shell** 命令；用 **exit** 返回 **vi**	
环境设置	**:set all**	显示 **vi** 的全部环境参数及它们的现行值	
	:set [**no**]**number**	设定要[不要]在屏幕左侧显示行号	
	:set [**no**]**ai**	设定新开行要[不要]继承前行的行首缩进	
	:ab V void *****	**V** 定为一替代词，在插入模或底线模，输入 **V<SP>**立即转换为 void *****	
	:map ^P "p	**<CTRL>P** 定为一命令：将先前存于缓冲区"p 中内容粘贴于本行下	
	上例中**^P** 是一不可打印字符；在定义时应打**<CTRL>V<CTRL>P**，用时则打**<CTRL>P**		

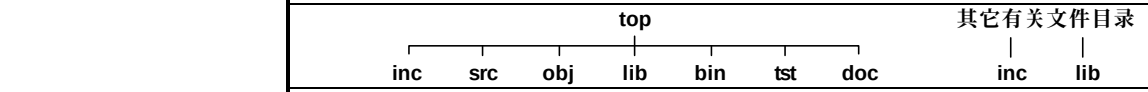
vi 初始化

$HOME/.exrc 文件示例

（如未设定环境变量 EXINIT，则 vi 自动按文件$HOME/.exrc 进行环境设置）

"file: .exrc	注释行。"号为行注释符
▼set用于设置环境参数	
set autoindent	新开行继承前行的行首缩进平移量
set shiftwidth=4	设定一个平移量为 4 个空格
set tabstop=8	设定一个制表符<TABLE>相当 8 个空格长
▼map用于定义宏命令，宏命令在命令模中充当新命令使用	
map (mn<'m'n	(，“ m 行 ”至本行左移一个平移量
map) mn>'m'n	)，“ m 行 ”至本行右移一个平移量
map ^K "py'm	<CTRL>K，将“ m 行 ”至本行的内容存入缓冲区"p
map ^X "pd'm	<CTRL>X，删去“ m 行 ”至本行存入缓冲区"p
map ^P "pP	<CTRL>P，将缓冲区"p 中内容粘贴于本行上
map ^A "pp	<CTRL>A，将缓冲区"p 中内容粘贴于本行下
map v :e #^M	v，转向先前编辑的文件
map V :w^Mv	V，先保存本文件，再转向先前编辑文件
map ^N :w^M:n^M	<CTRL>N，先保存本文件，再编辑下一个文件
map * mno */^[^hd0'mO/*^[^d0'njj	*，在“ m 行 ”前和本行后加"/* "和" */"各一行
map ^? o/*^M * ^M*/^[kA	<CTRL>*，先自动加三行/* 再从第 2 行末进入插入模 */
map _ :s/..*/&/^MI/* ^[A */^[	_，若本行非空，则在其首尾分别加"/* "和" */"
map # :'m,. s/^/# /^M	#，“ m 行 ”至本行的行首加"# "
map ^_ :'m,. s?^?// ?^M	<CTRL>/，“ m 行 ”至本行的行首加"// "
map ^\ :'m,. s/^.//^M	<CTRL>\，删去“ m 行 ”至本行行首的任意一个字符
map g :%s/	g，全文进行替换，待续被搜索模式和替换字串
map S :'m,. s/	S，从“ m 行 ”至本行进行替换，待续被替换模式和替换串
map ^O mn'mO# ifdef MODIFY^[^d0'no# else // MODIFY^[^d0o# endif // MODIFY^[O	<CTRL>O，在“ m 行 ”上加如下一行： # ifdef MODIFY 再回到本行下加如下两行，最后上开行进入插入模 # else // MODIFY # endif // MODIFY
map Q mn:%s/ *$//^M'n	Q，删去全文中行尾的空格
map q :g /^{/-1 p^M	q，显示全文中首列为{字符的前一行
map ^W :!pwd;df .^M	<CTRL>W，显示当前工作目录并列出相关盘区使用情况
map = :w^M:!make^M	=，先保存本文件，再执行 make 命令（不退出 vi）
map ^R :1,$!abc^M'm	<CTRL>R，用 abc 对全文（C 语言源程序）进行格式化
map ^L ?^{^M!%abc^M	<CTRL>L，用 abc 对包含本行的一个 C 函数进行格式化（abc 是一自编的 C 语言源程序的格式化程序）
▼ab用于定义替代词，在插入模或底线模字串搜索的输入过程中使用	
ab TT typedef	输入 TT<SP>，即转换为 typedef ，可继续插入
ab II # include	输入 II<SP>，即转换为# include ，可继续插入
ab ZZ {^M}^[O<SP><SP><SP>	输入 ZZ<SP>，变为：插入左花括号{，换行插入右花括号}，退出插入模，旋即又上开行插入 4 个空格，可继续插入

C 程序的文件系统目录结构



gcc，C 程序编程命令

gcc 命令的一般形式

	gcc	[-c]	[-I*incdir*]	[*src_file*]	[*obj_file*]	[*lib_file*]	[-o *output_file*]	[option]
一般：								
编译：	√	√	√	√			√	√
链接：	√				√	√	√	√

gcc	C 和 C++编译器
-I*incdir*	-I 指定头文件搜索路径 *incdir*，头文件（.h 文件）所在路径，如-I*../inc*
-c	完成编译，生成.o 文件
src_file	源程序文件名（.c 或.C 或.cc 或.cpp 或.cxx 文件）或预处理、汇编的输出文件名
obj_file	目标码文件名（编译生成的.o 文件）
lib_file	库文件名（.a 文件），库文件名的形式为 ***lib****name*.a，***lib*** 和.a 为惯用的前后缀
-o *output_file*	指定输出文件名。缺省：编译生成与源程序同名.o 文件，链接生成可执行文件 a.out
option	其它选项

gcc 命令中常用的其它选项

-v	在标准输出上显示编译步骤
-E	完成预处理，源文件为.c 或.C 或.cc 或.cpp 或.cxx 文件，预处理结果写至标准输出
-M 或-MM	预处理，输出与源文件同名的.o 文件所依赖的所有文件名（-MM 排除系统头文件）
-D*macro*[=*defn*]	增加宏定义，相当于在源程序中加入一行 #define *macro* [*defn*]
-U*macro*	取消源程序中已定义的一个宏 *macro*
-S	完成汇编，源文件为.c 或.C 或.cc 或.cpp 或.cxx 文件或.i 文件，生成.s 文件
-ansi	支持 ANSI 标准的 C 程序
-msupersparc	采用适于 SuperSparc cpu 的优化编译
-O 或-O2 或-O3	优化编译或进一步、再进一步优化编译。力图减小可执行文件的大小并减少执行时间
-Wunused	编译警告，指出未使用的变量
-Wuninitialized	编译警告，指出函数中的自动变量未赋值
-Wformat	编译警告，指出 scanf、printf 等函数调用中有格式错误
-Wreturn-type	编译警告，指出函数返回值与函数类型不符或无返回值
-Wall	多种编译警告的组合
-Werror	把编译警告视为错误，出现任何一个警告即中止编译
-g	调试选项，最终生成的可执行文件在运行时可用调试工具进行调试
-pg	调试选项，最终生成的可执行文件在运行时将附带生成运行情况分析文件 gmon.out
-L*libdir*	-L 指定库搜索路径 *libdir*，库文件所在目录，如-L*../lib*
-l*name*	-l 指定“库名”*name*，如-l*App*，*App* 加惯用前后缀得实际库文件名，即 ***lib****App*.***a***
man gcc	联机查询 gcc 的详细介绍

库

ar，库的生成与维护命令

ar -rv *aFile oFile …*	在 *aFile*（**.a** 文件）中添加或更新若干 *oFile*（**.o** 文件）并报告
ar -t *aFile* [*oFile …*]	列出 *aFile* 中全部或指定的 *oFile*
ar -d *aFile oFile …*	从 a*File* 中删去指定的 *oFile*

常用公用库的路径、库名

-L/usr/lib	一般系统库所在目录（默认路径）
-L/usr/local/lib	一般公用库安放目录（默认路径）
-L/usr/openwin/lib	Openwindow 图形系统库所在目录
-lm	数学函数库，全名为 **libm.a**，在**/usr/lib** 目录下
-lX11	X-window 图形库，全名为 **libX11.a**，通常在**/usr/openwin/lib** 目录下

make，程序或文件的维护、更新或重建

make 命令

make [**-f** *make_file*] [*macro*=*defn*] [*target*] …

-f *make_file*	指定 **make** 文件为 *make_file*，缺省为 **makefile** 或 **Makefile**（前者优先）
macro=*defn*	先将 *defn* 作为值赋予 **make** 文件中的宏 *macro*，再执行
target	执行 **make** 文件中的目标 *target*。缺省为 **make** 文件中的第一个目标

make 文件中的宏

macro **=** *value*	定义宏 *macro* 的值为 *value*
macro1 **= $(***macro***:**a=b**)**	由宏 *macro* 中字串 a 易为字串 b 派生出另一宏 *macro1*
macro2 **= $(***macro***:**a%b=c%d**)**	由宏 *macro* 中字串 a%b 易为字串 c%d 派生出另一宏 *macro2*，其中%代表除子串 a、b 和空格以外的任意字串

SRC **= $(wildcard ***.c**)**	宏 SRC 由通配符*****.c 构成	（**GNU** 版本的 **make** 支持）
SRC**:sh = echo ***.c	宏 SRC 由 **shell** 命令 **echo ***.c 构成	（**CCS** 版本的 **make** 支持）

make 文件中目标的依赖及其实现

target **:** *dependency*	目标及其依赖项
commands	实现本目标的命令组，命令数不限，行首需冠以**<TAB>**，可续行
.c.o :$(CC) $(CFLAGS) -c $<	由**.c** 生成**.o** 文件的编译命令（此为简约的后缀依赖规则）

make 命令的执行规则

第一目标：**make** 命令行中指定目标 *target* 或 **make** 文件中第一个目标为第一目标
派生目标：一依赖项若是本文件中的另一目标，则此依赖项为派生目标，形成待执行的目标链
执行规则：派生目标先行；若目标文件不存在或其依赖文件更新在后，则执行该目标规定的命令组

make 文件中动态宏及若干特殊字符

$@ 代表目标名	**$*** 截去目标名的目录与后缀	**%** 任意字符
$(@D) 目标名的目录名部分	**$<** 由目标名衍生的依赖文件	**** 在行尾表示续接下行
$(@F) 目标名的文件名部分	**$?** 更新了的依赖文件	**#** 在行首表示本行为注释

a**%**b : c**%**d 依赖项由目标名变换首尾而定	*@command* 执行命令 *command*，但不显示

执行多个无关目标

multitarget : t1 t2 t3	“目标”multitarget“依赖于”相互无关的子目标 t1 t2 t3
t1 **:** *dependency*	子目标 t1 及其依赖项
commands	实现目标 t1 的命令组

一份较完整的 Makefile

```
TOP          = ..                                          本 Makefile 所在目录的上层目录设为顶层目录
ALLSRC = $(wildcard *.c)或 ALLSRC:sh = echo *.c本目录下所有.c 文件名
MAINSRC      = app.c                                       含主程序的.c 文件名
SOURCES      = $(ALLSRC:$(MAINSRC)=)                       本目录下除$(MAINSRC)外所有.c 文件名
OBJPATH      = $(TOP)/obj/                                 .o 文件所在目录
OBJECTS      = $(SOURCES:%.c=$(OBJPATH)%.o)                全部.o 文件名，由相应.c 文件名转换而得
LIBRARY      = $(TOP)/lib/libApp.a                         库文件 libApp.a 的路径
LIBS         = $(LIBRARY) -lm                              需链接的有关库，此为 libApp.a 和标准数学库
TARGET       = $(HOME)/bin/app                             可执行文件 app 的路径，HOME 是环境变量
CC           = gcc                                         用 gcc 作为编译器
APP_INC      = $(TOP)/inc                                  一个头文件路径；可能有多个不同的路径
INCFLAGS     = -I$( APP_INC)                               所有头文件搜索路径；可以包括多个不同路径
CFLAGS       = $(INCFLAGS) -O $(DBG)                       编译条件
LFLAGS       = $(CFLAGS) $(MAINSRC) $(LIBS)                链接条件
MK_DIR       = if [ ! -d $(@D) ]; then mkdir $(@D); fi 命令宏：若目标所在目录不存在，则建此目录
$(TARGET)    : $(LIBRARY) $(MAINSRC)                       目标$(TARGET)，依赖库 libApp.a 和 app.c
        $(PRF) $(CC) $(LFLAGS) -o $@                        生成可执行文件的命令
$(LIBRARY) : $(OBJECTS)                                    库 libApp.a，依赖$(OBJECTS)中全部.o 文件
        @$(MK_DIR)                                          若库 libApp.a 所在目录不存在，则建此目录
        @ar -rv $@ $?                                       将更新的.o 加入或换入库 libApp.a 并报告
        @ctags $(SOURCES) $(APP_INC)/*.h                    生成索引文件 tags
debug        :                                             当 make 命令为 make debug，执行此目标
        @$(MAKE) -f Makefile DBG="-g -DDEBUG"               生成可供调试的可执行文件
purify       :                                             当 make 命令为 make purify，执行此目标
        @$(MAKE) -f Makefile PRF=purify DBG=-g              生成可供检查内存分配错误的可执行文件
clean        :                                             当 make 命令为 make clean，执行此目标
        @rm -rf $(OBJPATH) tags                             删除.o（和.d）所在目录及 tags 文件
$(OBJPATH)%.o: %.c                                         任何.o 文件，依赖同名.c 文件
        @$(MK_DIR)                                          若.o 文件所在目录不存在，则建此目录
        $(CC) $(CFLAGS) -c -o $@ $<                         编译同名.c 文件产生.o 文件
```

选择以下一种方式自动生成.o(.d)文件对.h 文件的依赖（分别为 GNU 和 CCS 版本的 make 支持）

```
$(OBJPATH)%.d:                                             任何.d 文件，可不指明依赖项
        @$(MK_DIR)                                          若.d 文件所在目录不存在，则建此目录
        @$(CC) -MM $(INCFLAGS) $*.c | sed -e '1 s?$*\.o?$(OBJPATH)& $@?' > $@
                                                            生成.d 文件，内含.o 和.d 文件的详细依赖
include $(OBJECTS:.o=.d)                                   将所有.d 文件的内容于此处包含进 Makefile
.KEEP_STATE:                                               生成.make.state，将被包含进 Makefile 尾
```

遍历多个目录执行指定目标

```
DIRS         = d1 d2 d3 d4          定义宏 DIRS，含若干子目录 d1 d2 d3 d4
DO_MAKE      = @for dir in $(DIRS); do (cd $$dir; $(MAKE) $@) done;
                                    定义宏 DO_MAKE，遍历诸子目录执行指定目标
t1 t2 t3     :                      三个目标采用“形式”相同的命令组，t1 为缺省的第一目标
        $(DO_MAKE)                   进入子目录执行 make，$@代表指定的执行的目标
t4           :                      其它目标，采用不同形式的命令组
```

xxgdb，程序调试工具

进入 xxgdb

xxgdb [*program*]	进入 **xxgdb**，*program* 为被调试的可执行文件名

xxgdb 的常用命令

命令	菜单按钮	命 令 说 明
	Source Listing	弹出源程序显示窗口
	Command Button	弹出命令按纽窗口
	Display Window	弹出变量显示窗口
file *name*	**file**	指定可供调试的可执行文件名 *name* 以备调试
list (*function* | [*file*:]*n*}		指定函数 *function* 或源程序 *file* 的第 *n* 行显示在源程序显示窗口
break ([*file*:]*n* | *function*)	**break**	设定源程序 *file* 的第 *n* 行或函数 *function* 为断点
tbreak ([*file*:]*n* | *function*)	**tbreak**	设定源程序 *file* 的第 *n* 行或函数 *function* 为仅停一次的断点
info break	**show brkpts**	显示所有设定的断点
cond *b cond*		在 *b* 号断点，仅当条件 *cond* 为真时程序才停
ignore *b n*		*n* 次经过 *b* 号断点不停
(**dis**able | **ena**ble) [*b1 b2 …*]		使 *b1*，*b2 …*号或全部断点不起或起作用
delete [*b1 b2 …*]	**delete**	删除 *b1*，*b2 …*号或全部断点
display *expr*	**display**	当程序到达断点时，在变量显示窗口显示表达式 *expr* 之值
info display	**show display**	显示需显示的表达式
undisplay [*d1 d2 …*]	**undisplay**	撤消 *d1 d2 …*号或全部需显示的表达式
run [*arglist*]	**run**	开始运行程序。*arglist* 为运行该程序所需的命令行参数
print *expr*	**print**	显示表达式 *expr* 的值
set *var* = *expr*		将表达式 *expr* 的值赋给变量 *var*
next [*n*]	**next**	执行源程序的下 *n* 行（缺省 *n* 为 1），不进入调用的函数，
step [*n*]	**step**	执行源程序的下 *n* 行（缺省 *n* 为 1），进入调用的函数
until *n*		继续运行，停于 *n* 行
cont [*n*]	**cont**	继续执行，停于以后的第 *n* 个断点（缺省 *n* 为 1）
finish	**finish**	继续执行直到退出正执行的函数
where [[-] *n*]	**stack**	显示函数调用栈，全部或内 *n* 层或外 *n* 层
up [*n*]	**up**	上升 *n* 层函数调用（缺省 *n* 为 1）
down [*n*]	**down**	下降 *n* 层函数调用（缺省 *n* 为 1）
quit	**quit**	退出 **xxgdb**
help [*class* | *command*]		列出命令分类或指定类别的命令简介或指定命令的详细说明
/（光标在源程序显示窗口）	**search**	弹出搜索窗口用于设置对源程序窗口进行字串搜索
e（光标在源程序显示窗口）	**edit**	弹出窗口用指定的文本编辑器对正在调试的程序进行编辑

xxgdb 的环境设置

setenv XXGDBWINEDIT '**xterm -fn** Screen-Bold14 **-bg** Black **-fg** Yellow **-bd** Red **-T** "xxgdb" **-e vi**'	设定 **xxgdb** 中 **edit** 命令弹出 **xterm** 窗口，使用 **vi** 编辑器
~/.gdbinit	**xxgdb** 的初始化文件。进入 **xxgdb** 时先执行此文件中的命令

gprof，程序运行分析工具

gprof *image_file* [*profile*]	显示程序 *image_file* 运行分析结果 *profile*（缺省名为 **gmon.out**）

学用 UNIX 4

csh 编程示例

csh 变量值的表示

$*var*	变量 *var* 的值	$?*var*	变量 *var*：存在为 1，否则为 0
$#*var*	变量 *var* 所含字数	$*var*[$*n*]	变量 *var* 所含第$*n* 个字的值
$*var*[*]	变量 *var* 所含的全部字的值	$*var*[*2-4*]	变量 *var* 所含第 2 至第 4 个字的值
$*var*[-$*n*]	变量 *var* 所含第 1 至第$*n* 个字的值	$*var*[$*n*-]	变量 *var* 所含第$*n* 至第末个字的值

csh 进程中特殊变量值的表示

$0	本命令文件名（含路径）	`basename $0`	命令名（$0 的文件名部分）
$$	本进程（父进程）号	$#argv	命令行参数个数
$*或$argv[*]或$argv	全部命令行参数的值	$*2* 或$argv[*2*]	第 2 个命令行参数的值

csh 变量赋值

set *var* [= *value*]	将 *value* 赋值给变量 *var*，变量 *var* 含单个字。若无具体值，则视 *var* 为逻辑变量
set *var* = (*v1 v2 v3*)	将 *v1 v2 v3* 赋值给变量 *var*，变量 *var* 含 3 个字（相当 C 语言中字符串数组）
set *var*[$*n*]= *value*	将含多个字的变量 *var* 所含第$*n* 个字的值改为 *value*
@ [*var* = *expr*]	将表达式 *expr* 的值赋给变量 *var*。单个符号@为显示所有 csh 变量

运算操作符及其优先级

(…)	结合	~	互补
!	逻辑非	* / %	乘、除、余
+ -	加、减	<< >>	左移位、右移位

< > <= >=	小于、大于、小于或等于、大于或等于
== != =~ !~	相等、不等、字符串与字串模式匹配、字符串与字串模式不匹配

&	按位与	^	按位异或	|	按位或
&&	逻辑与	||	逻辑或		

= *= += ++ --	等于、自乘、自加、加 1、减 1

条件表达式

"`*command*`" == *string*	命令 *command* 写至标准输出的信息等于字符串 *string*
$argv[1] =~ *.c	命令行第一个参数与*.c 相匹配；!~表示不相匹配
-[e][d][f][r][w][x][o][z] $*file*	文件$*file*：存在?是目录?是普通文件?可读写执行?属本人?文件名为空?
!($#argv)	命令行参数个数为 0

csh 的流程控制

条件执行

```
if (expr) command
```

条件分支

```
if (expr) then
  ...
else if (expr) then
  ...
else
  ...
endif
```

开关分支

```
switch (string)
case labe:
  ...
breaksw
……
default:
  ...
breaksw
endsw
```

顺序循环

```
foreach var (wordlist)
  ...
end
```

条件循环

```
while (expr)
  ...
end
```

转移

```
goto label
```

标号语句

```
label:
```

人机交互基本方式

echo -n "Input a number: "	提示输入一个数，**-n** 不换行，光标停在本提示后
set *var* **= $<**	接受从标准输入设备输入的一整行字符串赋给 **csh** 变量 *var*
echo The input is **$***var*	显示 **csh** 变量 *var* 的值

例 1. d2u，DOS 格式转 UNIX 格式

用　法：d2u dos_file …

功　用：将指定的若干 **DOS** 格式的文件 dos_file …转换为 **UNIX** 格式，若未指定文件，将提示命令用法；并提示输入待处理文件名，若文件名不为空，则再次执行本命令

# !/bin/csh	**csh** 脚本的首行首列必须是井号#
switch ($#argv)	开关语句，按命令行参数的个数进行分支
case 0:	命令行参数个数为 0 的情况
goto noargs	转标号语句 noargs
breaksw	分支完
default:	其它情况：
foreach file ($*)	**foreach** 循环，逐个将命令行参数记为 file
echo dos2unix: $file	告示即将对$file 作 dos2unix 处理
sed -e /^M\$/s/// $file >! _tmp	用 **sed** 命令将$file 内每行末的**^M** 删去后存入_tmp
\mv -f _tmp $file	强行用_tmp 复盖$file
end	循环完
breaksw	分支完
endsw	开关语句结束
exit	退出，结束本命令
noargs:	标号语句 noargs
echo Usage: `basename $0` dos_file ...	告示本命令的用法
echo -n "DOS files: "	提示输入若干待处理的 **DOS** 文件名
set files = $<	将键盘输入的文件名赋给变量 files
if ("$files" != "") $0 $files	如果$files 不空，则以其为命令行参数递归提交本命令

例 2. rn，更改多个文件名

用　法：rn str1 [str2]

功　用：将当前目录下所有文件名中所含字串 str1 删除或替换为字串 str2，实现多个文件改名。

# !/bin/csh	
if ($#argv == 0 || $#argv > 2) then	如果命令行参数个数为 0 或大于 2，则
echo Usage: `basename $0` str1 \[str2\]	告示本命令的用法
else	其它情况，即命令行参数个数为 1 或 2，则
foreach src (`echo **\***$1**\***`)	**foreach** 循环，逐个取含$1 的文件名为原名 src
set obj = (`echo $src | sed -e "/\$1/s//$2/"`)	将$src 中$1 替换为$2 形成新名 obj
echo -n "mv $src $obj ? [n] "	问改名否？（缺省是否）
set ans = $<	从键盘要回答
if ($ans =~ [Yy]*****) mv $src $obj	如果回答"y"或"Y"，则实行改名
end	循环完
endif	条件语句完

例 3. tg，文件打包压缩

用　法：tg source [[source ..]target]

功　用：将指定的若干目录或文件打包并压缩存入指定目录或当前目录。本命令采用管道方法同时执行 **tar** 和 **gzip** 命令。不产生中间的 **tar** 格式文件。在目的目录下生成若干加后缀**.tgz** 的压缩文件 source**.tgz**。在用 **tar** 命令打包时，剔除.o 和.a 文件，也不深入符号链接。

```
# !/bin/csh
if ($#argv == 0) then                          如果命令行参数个数为 0
  echo "Usage: tg source [[source ..]target] "告示本命令的用法
  exit 1                                        退出，令返回值为 1
else if ($#argv == 1) then                     如果命令行参数个数为 1
  set target = $cwd                             目的目录 target 就是当前目录
  @ flc = 1                                     用 flc 记待打包文件数，为 1
else                                           否则，即命令行参数个数大于 1 的情况
  if !(-d $argv[$#argv]) then                   如果最后一个命令行参数不是目录，则
    echo $argv[$#argv]\: no such directory        宣告最后一个命令行参数不是目录
    exit 2                                        退出，令返回值为 2
  endif                                         条件语句完
  set target = $argv[$#argv]                    目的目录 target 就是最后一个命令行参数
  @ flc = $#argv - 1                            用 flc 记待打包文件数，等于命令行参数个数减 1
endif                                          条件语句完
foreach file ($argv[-$flc])                    foreach 循环，每次将一待打包文件名记为 file
  set dir = `dirname $file`                     取待打包文件名的目录名部分记为 dir
  set base = `basename $file`                   取待打包文件名的文件名部分记为 base
  (chdir $dir; tar cfvFF - $base) | gzip -c -9 >! $target/$base.tgz
                                                借助管道，对待打包文件打包-压缩，写至目的目录
end                                            循环体完
exit 0                                         成功退出，令返回值为 0
```

例 4. gt，文件解压列表或复原（简化）

用　法：gt file [file …] [target]

功　用：将指定打包压缩的文件解压列表或在指定目录下解压复原

```
# !/bin/csh
if !($#argv) exit                              如果命令行参数个数为 0，则退出
if (-d $argv[$#argv]) then                     如果最后一个参数是目录，表示要复原，则做
  set target = $argv[$#argv]                    将最后一个参数记为 target
  @ flc = $#argv - 1                            用 flc 记待解压文件数，等于命令行参数个数减 1
  if ($flc == 0) exit;                          如果待解压文件数为 0，则退出
  foreach file ($argv[-$flc])                   foreach 循环，逐一取待解压文件名记为 file
    gzip -cd $file | (chdir $target; tar xfv -)   借助管道，对待解压文件解压-复原于目的目录
  end                                           循环完
else                                           否则，只做解压列表
  foreach file ($argv)                          foreach 循环，逐一取待解压文件名记为 file
    gunzip -c $file | tar tfv - | more            借助管道，对待解压文件解压、分屏显示内中目录
  end                                           循环完
endif                                          条件语句完
```

例 5. tcp，文件打包复制（简化）

用　法：tcp source [[source …] target]
功　用：将指定的若干目录或文件复制到另一目录。本命令借助管道，采用 **tar** 命令，一端打包，另一端复原。特点：1.剔除.o 和.a 文件；2. 不深入符号链接；3.文件日期不变。

```
# !/bin/csh
if !($#argv) exit                                  如果命令行参数个数为 0，退出
if ($#argv == 1) then                              如果命令行参数个数为 1，则
  set target = $cwd                                  目的目录 target 就是当前工作目录
  @ flc = 1                                          用 flc 记待复制文件数，为 1
else                                               否则，即命令行参数个数大于 1
  if !(-d $argv[$#argv]) exit                        如果最后一个命令行参数不是目录名，则退出
  set target = $argv[$#argv]                         目的目录 target 就是最后一个命令行参数
  @ flc = $#argv - 1                                 用 flc 记待复制文件数，等于命令行参数个数减 1
endif                                              条件语句完
@ n = 1                                            设循环变量 n 初值为 1
while ($n <= $flc)                                 while 循环，若$n 小于等于待复制文件数，做
  set dir = `dirname $argv[$n]`                      取一待复制文件名的目录名部分记为 dir
  set base = `basename $argv[$n]`                    取该待复制文件名的文件名部分记为 base
  (chdir $dir; tar cfFF - $base) | (chdir $target; tar xfvBp -)
                                                     对该待复制文件打包通过管道传至目的目录复原
  @ n ++                                             循环变量 n 增 1
end                                                循环结束
```

例 6. dcp，差异文件打包复制（简化）

用　法：dcp src_dir file …
功　用：指定一源目录和当前目录下的若干文件名，将当前目录下指定文件同源目录下同名文件比较。对发现有差异的文件，显示二文件的差别段落，并征询是否更新当前目录下的文件。

```
# !/bin/csh
if ($#argv < 2) exit                               如果命令行参数个数为小于 2，退出
if !(-d $1) exit                                   如果第一个命令行参数不是目录，退出
foreach file ($argv[2-])                           foreach 循环，第二个命令行参数起，逐一记为 file
  echo compare $file $1/$file …                      告示将比较文件$file 和源目录$1 下的同名文件
  cmp -s $file $1/$file                              用 cmp 命令比较$file 和$1/$file，-s 不报告任何信息。
  if ($status == 1) then                             如果状态参数 status 值为 1，表明二文件有差别，则
    diff $file $1/$file                                用 diff 命令显示二文件的相异段落
    echo -n tcp $1/$file . '? (y|q| [n])'              问是否要用 tcp 更改文件$file？（缺省为否）
    set ans = $<                                       从键盘要回答
    if ($ans =~ [yY]*)                                 如果回答"y"或"Y"，则
      tcp $1/$file .                                     用 tcp 更改当前目录下的文件$file
    else if ($ans =~ [qQ]*) then                       如果回答"q"或"Q"，则
      exit                                               退出，中止比较
    endif                                              条件语句完
  endif                                              条件语句完
end                                                循环完
```