|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数名 | 输入参数 | 返回信息 | 处理说明 | 关联操作 |
| GETATTR | FH：目标文件句柄 | type：文件类型  mode：文件权限掩码  nlink：硬链接数  uid：文件所属用户ID  gid：文件所属用户组ID  size：文件大小  used：文件实际使用的磁盘空间数  rdev：描述设备文件  fsid：文件系统标识符  fileid：文件系统的唯一标识文件编号  atime：最后访问时间  mtime：创建文件内容最后修改时间  ctime：文件属性最后修改时间 | 该函数用于获取指定文件的基本信息，包括权限、大小、所有权、读写时间等等 | 测试到的所有指令都调用了 |
| SETATTR | FH：目标文件句柄  new\_attributes：  mode  set\_it：是否更新  mode：更新值  uid  set\_it：是否更新  uid：更新值  gid  set\_it：是否更新  gid：更新值  size  set\_it：是否更新  size：更新值  atime  set\_it：是否更新  atime：更新值  mtime  set\_it：是否更新  mtime：更新值  guard：是否需要验证ctime | obj\_wcc：命令执行前后目标文件的属性信息（弱缓存一致性数据） | 用于修改文件权限、所属用户、所属用户组等 | chmod  chown  touch  vim |
| LOOKUP | what：查询的对象，包括：  what.dir：查询的目录句柄。  what.filename：符合命名规范的需要查询的文件名称。 | object：查找文件的句柄  obj\_attributes：文件的属性（与getattr接口返回参数一致）  dir\_attributes：查询目录的属性（与getattr接口返回参数一致） | 该函数用来查询一个目录下特定名字的文件，并返回对应的文件句柄。 | cp  mv  vim |
| ACCESS | object：验证权限文件句柄  access：需要验证的权限掩码 | access：表明请求中权限掩码验证通过的二进制掩码。  obj\_attributes：对应请求中的对象属性信息(getattr接口信息)。 | 该函数用于验证客户端操作权限，包括读取，删除，更改，添加文件内容或者目录条目，对于文件单独的权限为执行文件，对于文件夹的操作权限为查询文件，通过权限掩码进行请求判定 | sed  touch -p a/b/c.txt  vim  cp  echo  ln  mkdir -p  a/b  rm -r |
| READLINK | symlink：读取链接文件的句柄 | synlink\_attributes：对应请求中的对象属性信息(getattr接口信息)。  data:软件链接文件中存储的数据。 | 该函数从软连接读取数据，它返回的数据对于服务端来说是不透明的，也就是说无论是通过nfs3协议在客户端创建还是在服务端创建，软连接中的数据都不会被解释，只是进行简单的存储，所以他只能操作由nfs3创建的软连接。 | cat |
| READ | file：需要读取的文件句柄。  offset：开始读取的位置，如果读取的大小大于或等于文件的大小，则返回数据设置eof为trueq,count为0  count：读取的信息大小，必须小于等于设置rtmax，超过会造成数据损失。 | obj\_attributes：对应请求中的对象属性信息(getattr接口信息)。  count：读取的信息大小。  eof：是否完成读取。  data：读取的数据。 | 读取文件信息。 | cat  vim  cp |
| WRITE | file:需要写入的文件句柄。  offset:写入信息的位置，0从头开始写入。  count:写入数据的大小，0写入成功，报错是因为权限验证，写入的大小必须小于等于wtmax，写入大于设置可能造成数据丢失。  stable:  enum{  UNSTABLE=0;  DATA\_SYNC=1;  FILE\_SYNC=2;  }  FILE\_SYNC:服务器必须在返回结果前，将写入数据和所有文件系统的元数据提交落盘。  DATA\_SYNC：服务器在返回前必须将所有的数据和足够的元数据落盘并检索数据。  UNSTABLE:在回复客户端前，服务器可以自由地将数据和元数据任何部分落盘，包括全部落盘和都不落盘，无法保证未提交的数据是否或者何时落盘，只能保证不会损坏数据或更改vref，落盘速度不会比客户端请求慢。  data:写入的数据。 | file\_wcc：文件属性弱缓存一致性数据。  file\_wcc.before：写操作前的文件属性，包括：  file\_wcc.before.size：写操作前的文件大小。  file\_wcc.before.mtime：写操作前文件最后一次更改时间。  file\_wcc.before.ctime：写操作前最文件后一次属性更改时间。  linkdir\_wcc.after：写操作后的文件属性，getsttr接口信息。  count:写入文件数据的大小，可能比请求设置的写入的少。  committed:服务端返回的数据提交等级，如果提交了全部的数据和元数据设置为FILE\_SYNC，如果至少为DATA\_SYNC设置为DATA\_SYNC，其他设置为UNSTABLE，如果stable设置的是FILE\_SYNC则只能为FILE\_SYNC；如果设置的是DATA\_SYNC则返回可以为FILE\_SYNC,DATA\_SYNC；设置为UNSTABLE，则三种皆可设置，其他设置违反协议。  vref:用来在客户端验证在写操作和接下来的写或者提交操作之间服务器是否改变了状态，必须保证cookie在服务期间他一直存在，并且在多个服务实例之间唯一，否则可能造成数据丢失。 | 该函数用于写文件。 | echo  vim  cp |
| CREATE | where：待创建文件信息where.dir：文件所在目录文件句柄；  where.name：文件名  mode：创建模式  obj\_attributes：文件属性，当mode=GUARDED/UNCHECKED时存在（与SETATTR参数new\_attributes一致）  verf：验证信息，当mode=EXCLUSIVE时存在 | obj：新文件句柄  obj\_attributes：文件属性（与GETATTR返回一致）  dir\_wcc：文件所在目录弱缓存一致性信息 | 创建文件 | touch |
| MKDIR | where：待创建目录信息（与CREATE参数where一致）  attributes：目录属性（与SETATTR参数new\_attributes一致） | obj：新目录文件句柄  obj\_attributes：目录属性（与GETATTR返回一致）  dir\_wcc：所在目录弱缓存一致性信息 | 创建目录 | mkdir |
| SYMLINK | where：待创建文件信息（与CREATE参数where一致）  symlink\_attributes：属性（与SETATTR参数new\_attributes一致）  symlink\_data：软链源文件路径 | obj：新软链文件句柄  obj\_attributes：目录属性（与GETATTR返回一致）  dir\_wcc：所在目录弱缓存一致性信息 | 创建软链接 | ln -s |
| MKNOD | where：待创建文件信息（与CREATE参数where一致）  type：待创建待文件类型  device：当type=NF3CHR/NF3BLK时存在  pipe\_attributes：属性（与SETATTR参数new\_attributes一致），当type=NF3SOCK/NF3FIFO时存在 | obj：新文件句柄  obj\_attributes：目录属性（与GETATTR返回一致）  dir\_wcc：所在目录弱缓存一致性信息 | 创建特殊文件 | mkfifo |
| REMOVE | object：待删除文件信息（与CREATE参数where一致） | dir\_wcc：所在目录弱缓存一致性信息 | 删除文件 | rm  vim |
| RMDIR | object：待删除目录信息（与CREATE参数where一致） | dir\_wcc：所在目录弱缓存一致性信息 | 删除目录 | rm |
| RENAME | from:需要重命名的文件，包括：  from.dir:文件所在目录句柄。  from.name:符合文件命名规范的需更改的文件名。  to:新命名的文件，包括：  to.dir:新命名文件所在目录句柄。  to.name:符合文件命名规范的新文件名。 | fromdir\_wcc：源目录属性弱缓存一致性数据。  fromdir\_wcc.before：重命名前前的文件属性，包括：  fromdir\_wcc.before.size：重命名前源目录的大小。  fromdir\_wcc.before.mtime：重命名前源目录最后一次更改时间。  fromdir\_wcc.before.ctime：重命名前源目录最后一次属性更改时间。  fromdir\_wcc.after：重命名后源目录属性，getsttr接口信息。  todir\_wcc：目标目录属性弱缓存一致性数据。  todir\_wcc.before：重命名前的目标目录属性，包括：  todir\_wcc.before.size：重命名前目标目录的大小。  todir\_wcc.before.mtime：重命名前目标目录最后一次更改时间。  todir\_wcc.before.ctime：重命名前目标目录最后一次属性更改时间。  todir\_wcc.after：重命名后目标目录属性，getsttr接口信息。 | 重命名文件或者目录，必须保证服务端和客户端的原子性，并且重命名文件必须在同一个文件系统和服务下。 | mv |
| LINK | file：需要创建硬链接的文件句柄。  link：需要创建的硬链接的位置。其参数为：  link.dir：创建硬链接的目录句柄。  link.name：符合文件命名规范的名字。 | obj\_attributes：对应请求中的对象属性信息(getattr接口信息)。  linkdir\_wcc：文件属性弱缓存一致性数据。  linkdir\_wcc.before：创建硬链接操作前的文件属性，包括：  linkdir\_wcc.before.size：创建硬链接操作前的文件大小。  linkdir\_wcc.before.mtime：创建硬链接操作前文件最后一次更改时间。  linkdir\_wcc.before.ctime：创建硬链接操作前最文件后一次属性更改时间。  linkdir\_wcc.after：创建硬链接操作后的文件属性，getsttr接口信息。 | 该函数用于创建在同一个服务和文件系统下的文件硬链接 | ln |
| READDIR（弃用） | dir：读取的文件目录句柄。  cookie：cookie信息，第一次请求设置为0，接下来的请求中为服务端返回的。  cookieverf：cookie验证，第一请求设置为0，接下来得请求中为服务端返回的，必须和请求返回的cookie匹配。  count：服务端成功返回的数据的最大大小，服务端可返回任意小于它大小的数据。 | obj\_attributes：对应请求中的对象属性信息(getattr接口信息)。  cookieverf：cookie验证信息。  reply：读取的目录条目列表包括：  reply.entries:零或多个目录条目，包括：  reply.entries.\*entries:当前的条目，其包括：  reply.entries.\*entries.fileid:文件id。  reply.entries.\*entries.name:文件名。  reply.entries.\*entries.ookie:文件验证cookie。  reply.entries.\*entries.\*nextentry:下一个文件节点。  reply.entries.\*eof:是否最后一个条目，true是，false不是。 | 此函数用来读取目录条目信息，返回读取的目录条目链表 | ls  rm -r |
| READDIRPLUS | dir：读取的文件目录句柄。  cookie：cookie信息，第一次请求设置为0，接下来的请求中为服务端返回的。  cookieverf：cookie验证，第一请求设置为0，接下来得请求中为服务端返回的，必须和请求返回的cookie匹配。  dircount:返回的目录信息的最大大小，他不包括属性信息和文件句柄的大小（用户真正想获取的条目大小，如同READDIR的count）  maxcount：服务端成功返回的数据的最大大小，服务端可返回任意小于它大小的数据。 | obj\_attributes：对应请求中的对象属性信息(getattr接口信息)。  cookieverf：cookie验证信息。  reply：读取的目录条目列表包括：  reply.entries:零或多个目录条目，包括：  reply.entries.\*entries:当前的条目，其包括：  reply.entries.\*entries.fileid:文件id。  reply.entries.\*entries.name:文件名。  reply.entries.\*entries.ookie:文件验证cookie。  reply.entries.\*entries.\*nextentry:下一个文件节点。  reply.entries.\*entries.name\_attributes:文件属性信息（getattr信息）。  reply.entries.\*entries.name\_handle:文件句柄。  reply.entries.\*eof:是否最后一个条目，true是，false不是。 | 该函数用于读取文件系统目录的条目，相对于READDIR，每个条目多返回了文件属性信息（getattr）和句柄。 | ls  rm -r |
| FSSTAT | fsroot：需要获取信息的文件句柄 | 获取文件的基本信息obj\_attributes：getattr 接口信息。  额外的信息：  tbytes：文件系统总空间大小。  fbytes：文件系统剩余空间大小。  abytes：使用bytes计算的剩余空间信息，供通过rpc身份认证的用户使用。  tfiles：文件系统中文件插槽的总数，（在unix系统中通常对应配置的节点信息）  ffiles：文件系统中剩余文件插槽的数目。  afiles：共通过rpc验证的用户使用的空闲插槽数目。  inversec：文件系统更改的描述，用来设置文件系统在设置时间内不期望被更改，例如在不变系统下可以设置为最大的无符号整数，在变化频繁的系统下，设置为0，用户端可以使用它来进行缓存管理，是可变的并且可以在任何时候更改。 | 该函数用于返回文件系统可更改的状态信息 | du  rm -r |
| FSINFO | fsroot：需要获取信息的文件句柄 | obj\_attributes：对应请求中的对象属性信息(getattr接口信息)。  nfs协议使用的信息：  rtmax：服务端允许的最大读请求。  rtpref：期望的读请求大小，除非有明显的性能优化，否则应和rtmax一致。  rtpref：建议的读请求的大小的倍数。  wtmax：服务端支持的最大写请求。  wtpref：期望的写请求的大小，除非有明显的性能优化，否则应和wtmax一致。  wtmult：写请求大小建议的倍数。  dtpref：期望的READDIR请求的大小。  maxfilesize：文件系统最大的文件大小。  time\_delta：服务时间精度限制，当使用SETATTR建立文件时间应遵循的时间精度，当设置为{0，1}则精确到纳秒，{0，1000000}精确到微秒，{1，0}精确到秒。  properites：文件系统支持的特性二进制掩码：  FSF\_LINK：设置为1 ，则说明支持硬链接。  FSF\_SYNLINK：设置为1，支持软连接。  FSF\_HOMOGENEOUS：  如果设置为1,pathconf接口为文件系统中每个文件和目录返回的信息是一样的，设置为0，则在客户端需要在每个文件或者目录使用时单独请求pathconf。  FSF\_CANSETTIME：如果设置为1，当setattr请求时服务端会按照time\_delta设置时间，设置为0不会再请求时设置时间。 | 该函数用于返回文件系统不易更改的属性信息和nfs协议的服务接口限制信息 | mount |
| PATHCONF | object：需要获取信息的文件句柄 | obj\_attributes：对应请求中的对象属性信息(getattr接口信息)。  linkmax：一个文件的最大硬链接数目。  name\_max：文件命名的最长信息。  no\_trunc：如果设置为true则拒绝所有名字超过长度的请求，设置为false，则将名字截取为最大命名长度。  chown\_restricted：这是为true,如果不是超级管理员，则将拒绝所有更改文件用户或者用户组的信息的请求。  case\_insensitive：设置为true则服务不区分文件名大小写。  case\_preserving：设置为true，则在CREATE，MAKDIR，MKNOD，SYNLINK，RENAME，或者LINK 操作期间保存文件名大小写。 | 该函数用于返回文件配置信息，当FSF\_HOMOGENEOUS设置为1则所有导出的文件系统中文件和目录的设置一样 | mount |
| COMMIT | FH：目标文件句柄  offset：偏移量  count：数据量 | file\_wcc：文件弱缓存一致性信息  verf：验证cookie。客户端用于确定WRITE和COMMIT之间服务器是否重启 | 将服务器缓存数据落盘 | vim |

new\_attributes

dir\_wcc

before

attributes

size：操作之前的文件大小

mtime：操作之前的文件最后修改时间

ctime：操作之前的文件属性修改时间

after

attributes（与GETATTR返回一致）