

# Table of Contents

<b><u>Your Environment</u></b> .....	<b>1</b>
<u>Customizing Your Environment</u> .....	1
<b><u>Files</u></b> .....	<b>3</b>
<u>Description of Various Filesystems</u> .....	3
<u>Handling Your Files and Directories</u> .....	4
<u>Using Access Control Lists for File Sharing</u> .....	5
<u>Using 'Giveto' to Copy Files and/or Directories to Another User</u> .....	9
<b><u>Modules</u></b> .....	<b>11</b>
<u>Modules</u> .....	11
<u>Table of All Modules</u> .....	13
<b><u>Productivity Hints</u></b> .....	<b>16</b>
<u>Setting Up SSH Passthrough</u> .....	16
<u>X11 Forwarding</u> .....	21
<u>An Introduction to Virtual Network Computing (VNC) for Connecting to NAS</u>	
<u>High-End Computers</u> .....	23

# Your Environment

## Customizing Your Environment

Taking the time to customize settings in your NAS environment will help make your workflow more convenient and efficient. This article covers some common user customizations that can be included in your `.cshrc` file.

Note: If you want your customization to take effect for both interactive and non-interactive sessions, be sure to add them *above the line* (if it exists in your `.cshrc` file) that checks for the following prompt:

```
if (! $?prompt) exit #exit if not interactive
```

### Set Permission for New Files and Directories with `umask`

```
umask 077
```

Sets the permission of all new files and/or directories to be read/write/execute by owner only.

```
umask 037
```

Sets the permission to be read/write/execute by owner and readable by group members.

### Load Modules

For Columbia systems, if you want to use another set of modules (for example, intel-comp.11.1.056, mpt.1.22.0.0, scsl.1.6.1.0) other than the default, we recommend that you add the following to your `.cshrc` file:

```
module purge  
module load intel-comp.11.1.056 mpt.1.22.0.0 scsl.1.6.1.0
```

For Pleiades, there are no default versions. You can load the modules that you normally use in your `.cshrc`. For example:

```
module load comp-intel/11.1.072 mpi-sgi/mpt.1.26
```

See also: [Modules](#)

### Augment `$path`

If you normally use certain directory paths to find commands, scripts, or executables, you can add them to your path in `.cshrc`. We recommend that you at least add the current

directory "." to your path:

```
set path = ( $path .)
```

This removes the need to add "." in front of scripts and executables when running them in the current directory.

You can add other directories, such as the /bin directory under your home, to the path:

```
set path = ($path ~/bin .)
```

## Set Aliases

Aliases allow you to replace one string with another string. For example, if you add the following line in your `.cshrc` file, the command `ls -lrt` is used whenever you type `ls`:

```
alias ls 'ls -lrt'
```

## Set Environment Variables

If you need to use certain environment variables regularly, you can set them in `.cshrc`. For example:

```
setenv CFDROOT /u/your_nas_username/bin
```

# Files

## Description of Various Filesystems

Follow the links below to find information of each filesystem available to users:

- [Pleiades Home Filesystem](#)
- [Pleiades Lustre Filesystems](#)
- [Columbia Home Filesystems](#)
- [Columbia CXFS Filesystems](#)
- [Lou Home Filesystems](#)

# Handling Your Files and Directories

Follow these guidelines when handling your files and directories under the various filesystems:

- [Quota Policy on Disk Space and Files](#)
- [Policy on SUID/SGID Scripts](#)

# Using Access Control Lists for File Sharing

A common way to share files and/or directories with group members or others is to use the **chmod** command to change the permissions. Yet **chmod** has limitations, so using Access Control Lists (ACLs) may sometimes be your method of choice.

When you issue the command **chmod g+rx filename**, for example, all the members in your group (**g**) gain read (**r**) and search/execute (**x**) access to that file, as shown below:

```
% ls -l foo
-rw----- 1 zsmith s0101 9 Jun 10 12:11 foo

% chmod g+rx foo

% ls -l foo
-rw-r-x--- 1 zsmith s0101 9 Jun 10 12:11 foo
```

However, **chmod** does not allow you to select which members of your group or which specific individuals outside of your group can access your files/directories. For this, use ACLs, which provide a mechanism for finer-grain control of file sharing. There are two ACL commands:

- **setfacl** - set file access control lists

SYNOPSIS

```
setfacl [-bkndRLPvh] [{-m|-x} acl_spec] [{-M|-X} acl_file] file ...

setfacl --restore=file
```

A detailed usage explanation of **setfacl** and its options can be found via **man setfacl**. Among the options listed:

1. The **-m** or **-M** option lets you "modify" the ACL, where **-m** expects an ACL on the command line and **-M** expects an ACL from a file or from standard input
2. The **-x** or **-X** option removes the ACL entries
3. The **-R** or **--recursive** option applies operations to all files and directories recursively
4. The **--test** option allows you to test the effect of changing the ACL without actually changing it
5. The **-b** option removes all extended ACL entries except the base entries of the owner, group, and others

- **getfacl** - get file access control lists

SYNOPSIS

```
getfacl [-dRLPvh] file ...

getfacl [-dRLPvh] -
```

A detailed usage explanation of **getfacl** and its options can be found via **man getfacl**.

Note that **setfacl** operations are supported on all Pleiades, Lou and Columbia filesystems except the Columbia home filesystems.

Before you grant another user or group access to certain files or directories, make sure that access to the parent directory (where the files or directories reside) is also allowed.

## Example 1

To allow another user (jbrown) to have read/execute (**rx**) permission on a file (*foo*) and to view the ACL before and after an ACL change:

```
% ls -l foo
-rw----- 1 zsmith s0101 9 Jun 10 12:11 foo

% getfacl foo
# file: foo
# owner: zsmith
# group: s0101
user::rw-
group:---
other:---

% setfacl -m u:jbrown:rx foo

% getfacl foo
# file: foo
# owner: zsmith
# group: s0101
user::rw-
user: jbrown: r-x
group:---
mask: r-x
other:---

% ls -l foo
-rw-r-x---+ 1 zsmith s0101 9 Jun 10 12:11 foo
```

## Example 2

To remove all extended ACLs in Example 1 except the base entries of the owner, group, and others:

```
% setfacl -b foo

% ls -l foo
-rw----- 1 zsmith s0101 9 Jun 10 12:11 foo
```

```
% getfacl foo
# file: foo
# owner: zsmith
# group: s0101
user::rw-
group::---
other::---
```

### Example 3

Continuing from Example 1, to test the granting of read/execute (**rx**) access to another group (group id 24176) without actually doing it:

```
% setfacl --test -m g:24176:rx foo foo: u::rw-,u:jbrown:r-x,g::---,g:g24176:r-x,m::r-x,

% getfacl foo
# file: foo
# owner: zsmith
# group: s0101
user::rw-
user:jbrown:r-x
group::---
mask::r-x
other::---
```

### Example 4

To allow another user (jbrown) recursive access to a directory (**dir.abc** which contains a file *foo2*):

```
% ls -ld dir.abc
drwx----- 2 zsmith s0101 17 Jun 10 13:19 dir.abc

% ls -l dir.abc
total 0
-rw----- 1 zsmith s0101 0 Jun 10 13:19 foo2

% setfacl -R -m u:jbrown:rx dir.abc

% getfacl dir.abc
# file: dir.abc
# owner: zsmith
# group: s0101
user::rwx
user:jbrown:r-x
group::---
mask:r-x
other::---
```

```
% getfacl dir.abc/foo2
```

```

# file: dir.abc/foo2
# owner: zsmith
# group: s0101
user::rw-
user:jbrown:r-x
group:---
mask:r-x
other:---

% ls -ld dir.abc
drwxr-x---+ 2 zsmith s0101 17 Jun 10 13:19 dir.abc

% ls -l dir.abc
total 0
-rwr-x---+ 1 zsmith s0101 0 Jun 10 13:19 foo2

```

## Example 5

Continuing from Example 4, to recursively remove all permissions user jbrown for a directory:

```

% setfacl -R -x u:jbrown dir.abc

% getfacl dir.abc
# file: dir.abc
# owner: zsmith
# group: s0101
user::rwx
group:---
mask:---
other:---

% getfacl dir.abc/foo2
# file: dir.abc/foo2
# owner: zsmith
# group: s0101
user::rw-
group:---
mask:---
other:---

```

For more information on ACLs, read **man acl**.

# Using 'Giveto' to Copy Files and/or Directories to Another User

NAS's in-house developed `giveto` script is built on the use of Access Control Lists (ACL). It allows one user (the giver) to copy files and/or directories to a `/nobackup` directory of another user (the recipient).

`giveto` is installed under `/usr/local/bin` on Pleiades, Columbia and Lou.

In the example below, user `zsmith` gives a copy of his `dir.abc` directory on Pleiades to user `jbrown`. The steps describe the `giveto` command used by each of them, and the results.

1. User `jbrown` uses the command `giveto -i zsmith` to automatically (a) create an INCOMING directory (if it does not already exist) under her `/nobackup/jbrown` and (b) grant user `zsmith` read/write/execute permission on this directory.

```
pfe20:/u/jbrown% giveto -i zsmith
nobackup[1] = /nobackup/jbrown
```

```
pfe20:/u/jbrown% ls -ld /nobackup/jbrown/INCOMING
drwxrwx---+ 2 jbrown s0202 4096 Jun 14 12:18 /nobackup/jbrown/INCOMING
```

2. User `zsmith` uses the command `giveto jbrown dir.abc` to automatically (a) create a subdirectory called `zsmith_0` under `jbrown`'s INCOMING directory, (b) copy `dir.abc` to `/nobackup/jbrown/INCOMING/zsmith_0`, (c) grant user `jbrown` read/write/execute permission on `/nobackup/jbrown/INCOMING/zsmith_0`, and (d) send an email to user `jbrown` regarding the copy.

```
pfe20:/home1/zsmith> ls -ld dir.abc
drwx----- 2 zsmith s0101 17 Jun 14 12:21 dir.abc/
```

```
pfe20:/home1/zsmith> giveto jbrown dir.abc
setfacl -m u:jbrown:rwx zsmith_0
setfacl -m u:jbrown:rwx zsmith_0/giveto.log
setfacl -m u:jbrown:rwx zsmith_0/dir.abc
setfacl -m u:jbrown:rwx zsmith_0/dir.abc/foo2
path = /nobackup/jbrown/INCOMING/zsmith_0
total 12
drwxrwx---+ 3 zsmith s0101 4096 Jun 14 12:29 .
drwxrwx---+ 3 jbrown s0202 4096 Jun 14 12:29 ..
drwxrwx---+ 2 zsmith s0101 4096 Jun 14 12:21 dir.abc
-rw-rwx---+ 1 zsmith s0101 44 Jun 14 12:29 giveto.log
```

Note: If the directory `zsmith_0` already exists prior to this step, `zsmith_1` would be used instead.

3. User `jbrown` receives an email from user `zsmith` with a subject line "giveto files". They see that the directory `dir.abc` has been copied successfully. Even though the directory `/nobackup/jbrown/INCOMING/zsmith_0` is still owned by user `zsmith`,

user *jbrown* now has permission to read/write/execute files and directories under **/nobackup/jbrown/INCOMING/zsmith\_0**.

```
pfe20:/u/jbrown% ls -lrt /nobackup/jbrown/INCOMING
total 4
drwxrwx---+ 3 zsmith s0101 4096 Jun 14 12:29 zsmith_0

pfe20:/u/jbrown%ls -lrt /nobackup/jbrown/INCOMING/zsmith_0
total 4
drwxrwx---+ 2 zsmith s0101 4096 Jun 14 12:21 dir.abc

pfe20:/u/jbrown%ls -lrt /nobackup/jbrown/INCOMING/zsmith_0/dir.abc
total 4
-rw-rwx---+ 1 zsmith s0101 8 Jun 14 12:21 foo2

pfe20:/u/jbrown%getfacl /nobackup/jbrown/INCOMING/zsmith_0/dir.abc
# file: /nobackup/jbrown/INCOMING/zsmith_0/dir.abc
# owner: zsmith
# group: s0101
user::rwx
user: jbrown: rwx
group::---
mask: rwx
other::---
```

Read **man giveto** for more information.

The **giveto** script was created by NAS staff member Arthur Lazanoff.

# Modules

## Modules

A system called "modules" to centralize the location of licensed products from vendors or software from public domain is installed on all NAS HECC systems.

To use the modules commands, you have to do either one of the following first:

1. Source the following files in your **.cshrc** or **.profile**

In **.cshrc** (for csh users):

```
source /usr/local/lib/global.cshrc
```

In **.profile** (for bash users):

```
source /usr/local/lib/global.profile
```

2. In the shell that you want to use the module commands, do one of the following:

For csh users:

```
%source /usr/share/modules/init/csh
```

For bash users:

```
%. /usr/share/modules/init/bash
```

The following are useful module commands to remember:

### **module avail**

- Used to find out what modules are available.

### **module list**

- Used to list which modules are loaded in your environment.

### **module purge**

- Used to unload all loaded modulefiles.

### **module load *module\_name1 module\_name2 ... module\_nameN***

- Used to load the desired modules.

**module switch *old\_module\_name new\_module\_name***

- Used to switch between two modules.

**module show *module\_name***

- Used to show changes to the environment that will happen if you load *module\_name*.

## Table of All Modules

The table below shows the available software managed through modules on Pleiades and/or Columbia. To request installation of a software as a module, please send an email to [support@nas.nasa.gov](mailto:support@nas.nasa.gov)

Note that the name of a software module may contain:

- software name
- vendor name
- version number
- varieties such as what compiler and/or what library is used to build the software

For example:

- `comp-intel/11.1.072` represents the Intel Compiler version 11.1.072.
- `mpi-sgi/mpt.2.04.10789` represents the SGI MPI library version mpt.2.04.10789.
- `mpi-mvapich2/1.4.1/intel` represents the MVAPICH2 MPI library version 1.4.1 built with an Intel compiler.

Use the `module avail` command to see all the available versions and provide the full name of a module when you decide to load a module.

### Available Modules (as of 30 August 2010)

Software	Platforms	Function
Intel compiler	Pleiades/Columbia	Compiler
Intel mkl	Pleiades/Columbia	Math/Scientific Library
Intel mpi	Pleiades/Columbia	MPI Library
SGI mpt	Pleiades/Columbia	MPI Library
SGI scsl	Columbia	Math/Scientific Library
automake	Columbia	Makefile Tool
boost	Columbia	C++ Library
cpan	Pleiades	Comprehensive Perl Archive Network
cscope	Columbia	Source Code Browsing Tool
drm	Pleiades	X Window Library Tool
eclipse	Pleiades	Software Development Environment
emacs	Pleiades	Text Editor
fieldview	Pleiades/Columbia	Data Visualization and Analysis Tool
flex	Pleiades	Text Scanner Generation Tool
fluent	Pleiades	CFD Modeling Application
gaussian	Pleiades/Columbia	Quantum Chemistry Application

gcc	Pleiades/Columbia	GNU C/C++ Compiler
gd	Pleiades/Columbia	Images Creation Library
git	Pleiades/Columbia	Version Control System
glib	Pleiades/Columbia	Low-level Core Library
gmp	Pleiades/Columbia	Math Library
gnuplot	Pleiades/Columbia	Data Visualization Tool
grace	Pleiades/Columbia	Data Visualization Tool
grads	Pleiades/Columbia	Data Visualization and Analysis Tool
gridgen	Pleiades/Columbia	CFD Grid Generation Tool
gsl	Pleiades/Columbia	GNU Scientific Library
hcss	Pleiades/Columbia	Herschel Common Science System
hdf4	Pleiades/Columbia	I/O Library and Tools
hdf5	Pleiades/Columbia	I/O Library and Tools
idl	Pleiades/Columbia	Data Visualization and Analysis Tool
idn	Pleiades	GNU Libidn
imagemagick	Pleiades/Columbia	Image Tool
java-sdk	Columbia	Programming Language
jpeg	Columbia	Image Tool
jvm	Pleiades	Java Virtual Machine
libxml	Columbia	C Parser and Toolkit
lsdyna3d	Pleiades/Columbia	Finite Element Application
matlab	Pleiades/Columbia	Numerical Computing Environment and Programming Language
mlp	Columbia	Multi-Level Parallelism Library
mpfr	Pleiades	Multiple-Precision Floating-point Computations Library
mpich2	Columbia	MPI Library
mvapich2	Pleiades	MPI Library
ncarg	Pleiades/Columbia	Graphics Library for Scientific Data
ncl	Pleiades/Columbia	NCAR Command Language
nco	Pleiades/Columbia	netCDF Operators
netcdf	Pleiades/Columbia	I/O Library
octave	Pleiades/Columbia	Numerical Computations Language
paraview	Pleiades	Data Visualization and Analysis Tool
parmetis	Pleiades/Columbia	Math/Numerical Library
pdf	Columbia	PDF File Generation Library
perl	Columbia	Programming Language
petsc	Columbia	Math/Numerical Library
parallel netcdf	Pleiades/Columbia	Parallel I/O Library
png	Columbia	Portable Network Graphics Format

pyMPI	Columbia	MPI Program Development with Python
python	Pleiades/Columbia	Programming Language
ruby	Pleiades	Programming Language
svn	Pleiades/Columbia	Revision Control Application
swig	Pleiades/Columbia	Software Development Tool
tcl-tk	Pleiades/Columbia	Scripting Language
tecplot	Pleiades/Columbia	Data Visualization and Analysis Tool
texlive	Pleiades	TeX System Application
totalview	Pleiades/Columbia	Debugger
udunits	Pleiades/Columbia	Data Format Library
visit	Pleiades/Columbia	Data Visualization and Analysis Tool
xv	Pleiades	Images Display Application
xxdiff	Pleiades	Graphical File And Directories Comparator And Merge Tool
yaml	Pleiades/Columbia	Human-Readable Data Serialization Format
zlib	Columbia	Data Compression Library

# Productivity Hints

## Setting Up SSH Passthrough

**Summary:** By investing some time to set up SSH passthrough, you can make your future NAS logins and inbound file transfers easier and faster.

---

The SSH passthrough feature allows you to log into any system in the enclave by typing just one SSH command. *Without* passthrough, you have to log into an SFE, and then log into pfe[20-27], bridge[1-4], cfe2, or Lou[1-2]. *With* SSH passthrough you "pass through" the SFEs directly to a NAS high-end computing system where you will do most of your work.

After following the steps below, you can use SSH from your local host to log into a NAS high-end computing system and be prompted for only your SecurID passcode (your PIN plus fob tokencode) and password. The public key authentication is handled automatically, so you will not be prompted for the passphrase of your private/public keys.

### Before You Start

Before you are ready to set up SSH passthrough, you must already have gone through the initial steps of logging into the NAS HECC enclave, as outlined in [Security Overview](#). That is, you need to have done the following steps, in the order listed below.

- [Enabled your fob, and created your password and pin](#)
- Mastered [two-step login with password](#)
- Set up [public key authentication](#) (creating SSH Public/Private key pair, and copying SSH public key to SFEs)
- Mastered [two-step login with passcode](#)

### How To Set Up SSH Passthrough

At this point, you are only three steps away from streamlining all your future logins and inbound file transfers. You will be able to log in quickly to any host in the NAS HECC enclave, and you will be able to copy files from your local host to any NAS host without first copying the files to an SFE.

### Step 1: Copy OpenSSH Public Key to Hosts Inside the Enclave

Hosts inside the enclave use [OpenSSH](#), so you will need to copy the OpenSSH version of your public key to the hosts inside the enclave and place the key in your `.ssh/authorized_keys` file. *This must be done for every system inside the enclave to*

which you want to connect using SSH passthrough.

The following example uses `lou2.nas.nasa.gov` as the enclave host and `sfe1.nas.nasa.gov` as the secure front end.

Substitute your NAS username for `username@`. If your local host username and your NAS username are the same, you can skip the `username@`.

## Copy your OpenSSH Public Key

If you have done the Pre-Steps above, you already have your public key on one or more of the bastion front ends, `sfe[1-4]`.

**TIP:** If you don't have an `.ssh` directory on the NAS host (in this example Lou2), make sure that you create one (log in to Lou2 and issue the command `mkdir .ssh`) before issuing the `scp` command below. Otherwise, the command will copy the file `id_rsa.pub` to Lou2 with the filename `.ssh`.

On your local host, type:

```
your_localhost% ssh username@sfe1.nas.nasa.gov
```

On `sfe1`, type:

```
sfe1% scp .ssh2/id_rsa.pub username@lou2:~/.ssh
```

## Add Your OpenSSH Public Key to Your `.ssh/authorized_keys` File on the Enclave Host

On `sfe1`, type:

```
sfe1% ssh username@lou2
```

On `lou2.nas.nasa.gov`, type:

```
lou2% cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

If you get the error `"/u/username/.ssh/authorized_keys: No such file or directory"` after issuing the above command, you likely have `set noclobber`, which prevents you from overwriting files. You can use the command `unset noclobber` first to avoid this problem.

**WARNING:** The permission for the `authorized_keys` file must be set to 600 (issue the command `chmod 600 authorized_keys`). Group/others write permissions on `/u/username` and `/u/username/.ssh` are not allowed for public key authentication. Repeat this step for the other NAS hosts you plan to use, such as `bridge[1-4]` or a PFE. That is, add your public key to the `.ssh/authorized_keys` file on the other hosts.

## Step 2: Modify the .ssh/config File on Your Localhost

In your `~/ .ssh/config` file on your localhost, add the entries for the hosts inside the enclave you want to access. If you do not have a `~/ .ssh/config` file, create a new file called `config` in your `~/ .ssh` directory and add the entries. For your convenience, we provide a template.

Download the template `.ssh/config` (a text file named `config_nas.txt`). Move this template to your `.ssh` directory on your localhost and rename it to `config`.

The contents of this file are also shown below. In this template, `sfe1` is used; you can switch to `sfe2`, `sfe3`, or `sfe4` if you want to use a different secure front end for SSH passthrough. Replace `<NAS_login_name>` with your NAS username.

If you will be using this file to access additional hosts outside of NAS, see the note "Customizing Your Own .ssh/config" below.

```
Host sfe
# Replace sfel by sfe2, sfe3, or sfe4 if sfel is unavailable
HostName                sfel.nas.nasa.gov

Host sfe sfe?.nas.nasa.gov
Ciphers                  aes128-cbc,aes192-cbc,aes256-cbc,3des-cbc
ForwardAgent            no
MACs                    hmac-sha1

Host sfe sfe?.nas.nasa.gov dmzfs?.nas.nasa.gov sup*.nas.nasa.gov
LogLevel                info
ProxyCommand            none

Host pfe pfe-last pfe.nas.nasa.gov pfe-last.nas.nasa.gov
HostKeyAlias            pfe20.nas.nasa.gov
ProxyCommand            ssh -oCompression=no sfe /usr/local/bin/ssh-balance %h

# Add additional hosts to the list below as needed
Host *.nas.nasa.gov lou lou? cfe? pfe? bridge? sfe pfe pfe-last
ForwardAgent            yes
HostbasedAuthentication no
Protocol                2
ProxyCommand            ssh -oCompression=no sfe /usr/local/bin/ssh-proxy %h
ServerAliveInterval    10m

# Replace <NAS_login_name> with your NAS username
User                    <NAS_login_name>

# Enabling compression may improve performance for slow connections
#Compression            yes

# Uncomment this line if you are using OpenSSH 4.7 or later
#MACs                   umac-64@openssh.com,hmac-md5,hmac-sha1
```

## Customizing Your Own .ssh/config File

If you use your `.ssh/config` file for accessing both NAS systems and systems at other sites, you can add entries at the top of the above template. The entries take the form:

```
Host hostname
ProxyCommand ssh username@hostname.nas.nasa.gov /usr/local/bin/ssh-proxy hostname
```

*Hostname* is the name of the host you want to access. It can be abbreviated as *hostname* (such as `pfe20`) or can be the fully-qualified domain name (such as `pfe20.nas.nasa.gov`). Note that using `bbftp` requires the fully qualified domain name, so it is a good idea to include both.

## Step 3: Set Up SSH Agent

The `ssh-agent` program holds and manages the private key on your local host and responds to key challenges from remote hosts. The private key is not initially stored in the agent and is added through the `ssh-add` program.

Typically, `ssh-agent` is started at the beginning of an X session or a login session, and you provide your passphrase to unlock your private key for this originating session. For any SSH connection to a remote host (for example, `sfe1`) made from this original session, the `ssh-agent` remembers your private key and will respond to challenges automatically without prompting you to type in your passphrase again.

How SSH passthrough works: If you want to use SSH to connect from the first remote host (for example, `sfe1`) to a second remote host (for example, `pfe20`) and possibly from the second remote host to a third remote host, a feature called **agent forwarding** allows a chain of SSH connections to forward all the key challenges back to the original agent, thus eliminating the need for using a password or public/private keys for these connections.

Note: In order for agent forwarding to work, your public key must be installed in all the remote hosts that you intend to connect to. See Step 1, above.

## Setting up SSH Agent on UNIX or LINUX Systems

If you use `csh` or `tcsh`, to launch `ssh-agent`, type the following command:

```
your_localhost% eval 'ssh-agent -c'
```

Or, if you use `sh` or `bash`, to launch `ssh-agent`, type the following command:

```
your_localhost% eval 'ssh-agent -s'
```

To add your private key to **ssh-agent**, type the following command:

```
your_localhost% ssh-add private_key
```

**Example:**

```
your_localhost% ssh-add ~/.ssh/id_rsa
```

```
Enter passphrase for /Users/username/.ssh/id_rsa: type your passphrase
```

Setting up SSH Passthrough may be complicated, but it is worth doing to save time in the future.

## X11 Forwarding

To run X applications (such as **xclock**, **emacs**, **totalview**, etc) on the X client host (for example, a NAS computer such as **pfe20**, etc.) and display them back to an X server (such as your localhost), the simplest way is to use SSH X11-Forwarding.

If you are using a NAS-supported workstation or compute server, X11-Forwarding is already set up for you. The following command activates SSH X-11 Forwarding automatically:

```
your_localhost% ssh hostname.nas.nasa.gov
```

Most modern SSH client software support this option (for example, Cygwin, TeraTerm, PuTTY, Unix, and Linux). To use SSH X11-Forwarding, the SSH server-side daemon (**sshd**) configuration file must contain the entry:

```
X11Forwarding yes
```

to support the forwarding capability. All NAS-supported hosts (including all workstations and compute servers) honor this setting in the default **sshd\_config** file (for High End Computing systems and Linux systems, it is **/etc/ssh/sshd\_config**; for Mac systems, it is **/etc/sshd\_config**) For other non-NAS machines, ask your system administration to set this in the **sshd\_config** file.

In addition to setting **X11Forwarding yes** on the SSH server side (for example, a NAS machine), it is recommended that **ForwardX11 yes** is also set in the **ssh\_config** file on the SSH client host (for example, your localhost). By default, NAS and HEC system configurations will enable these settings for both client and server. If **ForwardX11 yes** is not set in the **ssh\_config** file by the system administrator of your localhost, you can set it in your **~/.ssh/config** file or use the **-X** option of SSH.

Other parameters related to the performance of X11-Forwarding are handled by the NAS-recommended **ssh\_config** file. If you are on a NAS-supported system, no action is needed in setting these parameters yourself. If your localhost is not supported by NAS and you would like to get configuration ideas, you can look at **/etc/ssh/ssh\_config** on any NAS High-End-Computing systems (such as **cfe2**, **pfe[20-27]**, **bridge[1-4]**).

## Example

To run an X11-based application, for example, **xclock**, on **pfe20.nas.nasa.gov** and have it displayed on your localhost, do the following:

```
your_localhost% ssh pfe20.nas.nasa.gov
```

```
-----  
* * * W A R N I N G W A R N I N G * * *
```

```
U.S. GOVERNMENT COMPUTER
If not authorized to access this system, disconnect NOW.
```

```
.....
```

```
-----
```

```
** PFE20 **
```

```
.....
```

```
pfe20% xclock
```

To run an X11-based application on a NAS-supported desktop, follow the example below:

```
your_localhost% ssh sfel.nas.nasa.gov
sfel% ssh a_nas_desktop
a_nas_desktop% xclock
```

If **ForwardX11 yes** is not set in either the **ssh\_config** file or the **~/.ssh/config** file of your localhost, use:

```
your_localhost% ssh -X hostname.nas.nasa.gov
```

**WARNING:** The SSH daemon sets the **DISPLAY** environment variable by itself. DO NOT RESET it to point to display zero (for example, **setenv DISPLAY your\_localhost:0**), otherwise SSH X11-Forwarding will not work.

**WARNING:** For each new login, the **.Xauthority** file gets updated. If you are over your quota, this file cannot be updated and you get the error:

```
/usr/X11R6/bin/xauth: error in locking authority file
/u/username/.Xauthority
```

X11-Forwarding will not work if you receive this error.

# An Introduction to Virtual Network Computing (VNC) for Connecting to NAS High-End Computers

## Columbia Phase Out:

As of Feb. 8, 2013, the Columbia21 node has been taken offline as part of the [Columbia phase out process](#). Columbia22-24 are still available. If your script requires a specific node, please make the appropriate changes in order to ensure the success of your job.

Developed by ATT England, VNC provides a means to reduce X11 overhead on high-latency networks such as the Internet. In practical terms once a VNC session is underway latencies are on the order of seconds rather than minutes. VNC can make remote X11 applications useful instead of being tedious and non-productive.

The principle of operation involves a host server process (for example, Xvnc on Pleiades or Columbia at NAS) that communicates with X11 applications running on Pleiades or Columbia. The host server process translates the X11 images into something akin to an MPEG style video for display by a remote desktop/laptop's viewer client. Xvnc transmits images and image updates using a low overhead protocol to the user's viewer client.

## Security and Firewalls

In the NAS environment VNC traffic is carried by a SSH tunnel much like SSH is used to tunnel X11 traffic. Using an SSH tunnel provides security because SSH encrypts tunnel traffic in both directions. If a user is already using SSH, then VNC traffic will find its way to/from NAS over current connections and through current firewalls. There is no need for any additional communication updates/authorizations.

## Where is the VNC software?

The NAS Pleiades and Columbia systems are running with some versions of Linux. All the necessary VNC software is installed in `/usr/X11R6/bin`.

On a user's desktop/laptop, there is no need to run a X11 server since in the VNC environment all the X11 work is done on the Pleiades front-end systems (`pfe[20-27]`, `bridge[1-4]`) or the Columbia front-end (`cfe2`). However, a VNC client viewer is needed and it may have to be downloaded depending on whether this remote computer is running Linux, Mac or Windows. The client may already be installed in many Linux distributions and on recent versions of Mac OS X.

If you have a NAS supported desktop/laptop system, please note that:

- For NAS supported Linux workstations, a VNC client viewer (RealVNC version 4.1.2) should have been installed under `/usr/bin/vncviewer`
- For NAS supported Mac systems running the Snow Leopard OS, a VNC client called "Chicken of the VNC" (version 2.0b4) should have been installed under the `/Applications` directory

If you have a Mac which is not supported by NAS, you can download "Chicken of the VNC" (version 2.0b4) from <http://sourceforge.net/projects/cotvnc/>

If you have a Windows desktop system, you can download free VNC clients from:

- <http://www.realvnc.com/products/free/4.1/download.html>
- <http://www.tightvnc.com/download.php>
- <http://www.uvnc.com/download/index.html>

Ask your local system administrator for help on installing the VNC client software.

## Steps to a VNC session

The steps described below are not the only way in establishing a VNC session. However, they should prove to be more convenient in the sense that you do not have to manually find an available display number to use.

`cfe2` is used below as an example. You can substitute `cfe2` with `pfe[20-27]` or `bridge[1-4]` if you want to establish a VNC session on a Pleiades front-end.

### Step 1: SSH into cfe2

Starting a VNC connection/session is a matter of using SSH or some other SSH capable client to connect to `cfe2`. VNC is much easier to use if SSH Passthrough on your localhost has been set up.

Note that in your `.ssh/config` file on your localhost, you do not need to enable SSH X11 forwarding, but you do need **ForwardAgent yes**.

Once SSH Passthrough is set up properly, you can establish a SSH connection from your localhost to `cfe2`.

```
localhost% ssh cfe2
cfe2%
```

### Step 2: Invoke vncserver Command on cfe2

**vncserver** is a script that starts/stops/kills the actual VNC server Xvnc.

The first time you invoke **vncserver** on a server, you will be prompted to create a password for VNC. This password should be up to 8 characters in length. If you create a password longer than 8 characters, it will be truncated to the length of 8. This password is encrypted and saved in the **\$HOME/.vnc/passwd** file on the server. Once this is done, you won't be prompted for a password on the server when invoking **vncserver** for subsequent VNC connections.

```
cfe2% vncserver -localhost
```

You will require a password to access your desktops.

Password: <--- type in a password of your choice

Warning: password truncated to the length of 8.

Verify: <-- retype your password

New 'X' desktop is cfe2:25

```
Creating default startup script /u/username/.vnc/xstartup
Starting applications specified in /u/username/.vnc/xstartup
Log file is /u/username/.vnc/cfe2:25.log
```

There are a few options to the **vncserver** command, such as **:display** (for setting the display number), **-geometry** (for setting the desktop width and height in pixel), etc. The **-localhost** option shown in the above example is a local security option that you should use all the time. It must appear as the last option or it won't get processed.

Similar to an X11 session, a VNC session uses a display number. If not supplied, the **vncserver** searches over the valid range from 0 to 99 and assigns the next free display number for your session. In the above example, a display number of 25 is assigned.

### Step 3: Create a SSH Tunnel from Your localhost to the Server

The next step is to create a SSH tunnel from your localhost to the server. This is done by first escaping into an SSH sub-shell and specifying a local client's port number and a server's port number to use. The default SSH escape characters are **<return> ~C** (upper case 'C'). If you do not get the SSH prompt, repeat the **<return>~C**.

```
cfe2% ~C
ssh> -L 59xx:localhost:59xx
Forwarding port.
```

At the SSH prompt, provide a local client port and a remote server port. VNC by default uses TCP port 5900+xx. Thus, it is common to provide the value 59xx for both the local client port (the number before *localhost*) and server port (the number after *localhost*). For example, the number 5925 can be used for both. If this number does not work, other

numbers can be used, for example, 5825.

Note that the client port number and the server port number do not need to be the same. Some may suggest using a very high client port number such as 22222 or 33333 since high port numbers are less likely to be reserved for other purposes. For example:

```
cfe2% ~C
ssh> -L 22222:localhost:5925
Forwarding port.
```

The maximum number allowed for the client port is 65535. Avoid using the local port numbers 0-1024 (root privilege required), 5900 (for Mac systems, reserved for some Apple remote desktop products), and 6000-6063 (reserved for local X window server). Use the **netstat -an** command to check what local port numbers have been used:

```
localhost% netstat -an | less
tcp46      0      0 *.5900          *.*          LISTEN
tcp4       0      0 *.22           *.*          LISTEN
```

The above example shows local ports 5900 and 22 are in use and should be avoided.

## Step 4: Start the VNC Viewer Application on Your localhost

- If your localhost is a Mac and you have "Chicken of the VNC" installed, launch it. Open the Preferences panel from the "Chicken of the VNC" menu and select the Performance tab. Make sure the "Frontmost Connection" slider is not at its highest setting. If it is, move it down one notch. Close the Preferences panel. Now, open a new connection using the "New Connection" item from the "Connection" menu.

In the popup window "Connect", enter *localhost:22222* in the Host field (if your local port number is 22222 from Step 3), and your VNC password in the Password field. Then click on the "Connect" button.

- If your localhost is a Linux system, do:

```
localhost% vncviewer localhost:localhostnumber
You should get a password prompt. Enter your VNC password that you created on
the server.
```

The *localhostnumber* is the one you use in step 3. For example, if you choose 22222 as your local port, do:

```
localhost% vncviewer localhost:22222
```

If everything goes well, the Xvnc server will send a X11 window manager display to your localhost that will appear as an xterm in the viewer's window.

The default window manager is TWM, and there are a couple other window managers to choose from under `/usr/X11R6/bin`, such as FVWM, MWM, etc. The KDE window manager, available under `/opt/kde3/bin`, provides a GUI view of a user's files and includes a few useful tools. To use a non-default manager, for example KDE, modify your `HOME/.vnc/xstartup` file on the host where you start `vncserver` as follows:

```
#twm &
/opt/kde3/bin/startkde &
```

Be aware that the KDE window manager needs more memory. For Pleiades users, it is recommended that you use the front-end nodes `bridge1` and `bridge2` instead of `pfe[20-27]` if you want to use KDE for your VNC sessions.

You can also change the size and position of the xterm in your viewer's desktop by changing the values in the following line of the `HOME/.vnc/xstartup` file on the host where you start `vncserver`. For example:

```
xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
```

This specifies an xterm which is 80 characters wide, 24 characters high, at a position (10 pixels, 10 pixels) from the upper left corner of the VNC viewer's desktop.

**TIP:** The modifications to the `xstartup` file only take effect for a new VNC connection.

You will need to stop the existing VNC server and start a new one.

The window manager's xterms is running on `cfe2` itself. From this xterms, you can do tasks that you normally do on `cfe2`, for example, start an X application or `ssh` to other NAS systems. PBS jobs can also connect to a VNC session when the user provides the `DISPLAY` environment variable to the job. Specifically, in the xterm in the viewer's window, submit an interactive PBS job and set the `DISPLAY` variable to `vncserver_hostname:display_number` before starting an X application:

```
cfe2% qsub -I -lncpus=4,walltime=1:00:00
qsub: job 1030046.pbs20.nas.nasa.gov ready
PBS(4cpus)columbia21> setenv DISPLAY cfe2:25
PBS(4cpus)columbia21> xclock
```

## Step 5: Shut Down the Server When You are Done with the VNC Session

On each VNC server, there are a limited number of VNC sockets available. At the end of a session, be sure to exit the VNC application on your localhost so that others can use the sockets. In the terminal window where you started up VNC, use the following command to clean up a few temporary socket files `vncserver` had created.

```
cfe2% vncserver -kill :xx (supply the original display number)
```

For example:

```
cfe2% vncserver -kill :25  
Killing Xvnc process ID 3435054
```

**WARNING:** Don't manually kill `vncserver`. Doing so will leave lock and socket files (for example, `/tmp/.X11-unix/X25`, `$HOME/.vnc/cfe2:25.pid`, etc.) on the server.