How do you get an account?

Getting a GACRC account is a two-step process:

- 1. Adding the user to your lab in Chematix
- 2. Assigning an account to the user the RCC (GACRC) web portal. For help with these processes, feel free to contact Greg Derda - (542-0188, derda@uga.edu).

Instructions for step 1 (also found on the GACRC website at http://rcc.uga.edu/account_mgmt/subpages/user_mgmt_db/)

- Go to Chematix website
- log in
- click on 'Resource Management'
- click on 'View my locations' under the 'Manage Locations' heading
- click on the lab, which will take you to a Lab Summary page
- scroll to the bottom of the Lab Summary page
- click on 'Manage Lab Personnel'. This will bring you to a page called 'Assign Personnel to Laboratory'
- click on 'Search for Personnel to Assign'. This will bring you to the 'Search for User' page.
- search for a user. A list will pop up with users who meet your criteria.
- click the radio button next to the user you wish to add, and then click 'Select User'
- You will be returned to the user page and see that the user selected is now in the list of lab users.
- If you have no more changes to make to lab personnel, click the 'Submit' button at the bottom of the 'Assign Personnel to Laboratory' page
Instructions for step 2 (no documentation for this currently – there may be a lag time of a few hours while the Chematix database adds the new lab user to the lab in the RCC Web Portal)

- log in to the RCC Web Portal
- click on ‘Computing Lab User Account Permissions’
- select your computing lab on the next page. A list of accounts connected to that account will appear. (I selected the 00-00-RC005-000 account for this demo.)
- select the account that you wish to add the user to. A list of your lab users will appear below the account, with boxes checked next to the user who have access to that account.
- check the box next to the lab user you would like to give permission to use that account (I checked Katharine Korunes for the purposes of this demo.) Click on the submit button at the bottom of the list. Once the screen refreshes, you will see the box next to their name is now checked.
UNIX Is An Operating System (OS)

- UNiplexed Information and Computing System - UNICS
- Released in 1969
  - before Graphical User Interfaces (GUI), all typing
- Academic OS that you already use
  - Core tools (BLAST, FASTA, etc) written for UNIX
  - The Web (mostly) is built on UNIX
- Many variants
  - Mac OS X is UNIX-driven
  - Can be different, but in general, share a similar operation
UNIX Is Powerful and Adaptable

- Simple Input/Output (I/O)
- Program in UNIX (without PERL or others)
- Multitasking – runs many programs at once
- Multiuser – many users can log in at once
- Network-ready – your machine, next to you, the other side of the world with equal ease
  - Assuming you have access

UNIX: Basic Components

- Kernel - master control program
  - Resource allocation, file system and more
  - In the background
- Environmental Variables – global options in the UNIX OS
  - Can be changed
- Shell (sh) – This is the terminal
  - Command line interpreter and user interface (UI)
  - sh, bash, csh, tcsh, ksh, zsh (more)
  - Similar, different, can cause problems

Open a Terminal

- Applications/Utilities/Terminal
  - This is how we will use UNIX to interact with your machines and your accounts on other machines (GACRC)

UNIX: The Terminal

- Multiple windows, multiple tabs
- Modify appearance
  - Size, color, text size
- Inspector (Shell/Show Inspector)
- Preferences/Settings/Pull down under themes/restore defaults

UNIX: The Terminal

- Program for I/O and interaction with system
- Prompt – where commands are entered
  - Different symbols
  - ‘$’ in lectures
- Shell is ready for a command
  - Prompt will not be there if a command is running

A word about data flow

You must tell the computer what do do, ALWAYS

- How does data get in (STDIN)?
  - From the keyboard
  - From a file used as input
  - From the output of another program that generates a result

- Where does data go (STDOUT)?
  - To the screen. CAUTION: this is the default. (nice to look at but not so useful)
  - To a file. You must tell the computer to do this with a redirect command like “>”
  - To another program. You must tell the computer to do this with a redirect command like “|” pipe.

- There is another kind of data – error messages (STDERR)
  - They go to the screen or
  - They go to a file

STDIN  ↓
STDERR  ↓
STDOUT  ↑
**Slide Conventions**

- **Command Line**
- **Common Errors**
- **Populate Cheat Sheets with these**

**UNIX: Entering Commands**

- Type a command and hit ‘Enter’
- **Print Working Directory – $: pwd**

**Try It!**

**UNIX: File Hierarchy**

- Starts at ‘root’
- Files and folder are found on ‘file paths’
- **pwd** showed you the full file path to your ‘Working’ or ‘Current’ directory (same thing)
- Compare to finder
- Paths can get complicated
  - `/Users/Jeremy/Documents/Lab/API/SYN/SYN2/Circos/Synteny/Blah/Blah/…`
  - Directories above = ‘parents’
  - Directories below = ‘subdirectories’

**Organization**

<table>
<thead>
<tr>
<th>Specific to Global</th>
<th>Global to specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jessica Kissinger</td>
<td>/USA/30622/GA/Athens/</td>
</tr>
<tr>
<td>500 D.W. Brooks Dr. #370</td>
<td>UGA/Coverdell/</td>
</tr>
<tr>
<td>Coverdell Center</td>
<td>500BrooksDr/370/</td>
</tr>
<tr>
<td>University of Georgia</td>
<td>Kissinger</td>
</tr>
<tr>
<td>Athens, GA, 30622</td>
<td>USA</td>
</tr>
</tbody>
</table>

**UNIX: File Hierarchy**

- So where are you?
  - **pwd** gave you a file path
  - Where is that?
  - Change Directory (cd)
  - Demo
    - $: cd /
      - ‘/’ is root, the ‘top’ of all file paths
      - Applications, Users, Volumes
    - $: pwd
    - $: cd original/filepath
      - Back where you started!
      - Trace in Finder
Remote Login

- Secure Shell (ssh), like Telnet, but encrypted
- Allows a GUI via X Windows (more today)
- Works on all platforms
- Must have an ssh client installed on both machines
  - OpenSSH is the client on your machine
  $: man ssh

Remote Login

- Mac, UNIX or Linux machine

  • Using your machine address
  • Using Fully Qualified Domain Name (FQDN)
    $: ssh username@zcluster.rcc.uga.edu
    – Username=bin6005  gnbcg=everything before the first '.' on machine address
  • Using IP Address
    $: ssh username@128.192.###.##
    – '###.##'=everything after first '.' on machine address
  • Are you sure you want to continue connecting (yes/no)?
  – Yes
  • Enter pwd
  $: logout #when you are done

scp - you can take data with you, or go get it, or put it somewhere else...

- Secure Copy (scp)
- Copy files and folders from one machine to another
- Must have ssh client installed on both machines
- More secure than file transfer protocol (ftp)

scp: format depends on action

- Here to there
  $: scp local_file user@target.machine:path/to/

- There to here (current directory)
  $: scp user@source.machine:path/from .

- There to here (other specific location)
  $: scp user@source.machine:path/from path/to
**scp**

- `-r` argument will move entire directories

```bash
$ scp -r local_folder jdebarry@rcluster.rcc.uga.edu:
```

- Often helpful to have two terminal tabs open, one for each location

**UNIX: Orienteering**

- Home Sweet Home (`~`)
- Working directory when you open terminal
  - This is your ‘special place’, you are in charge
  - Can’t necessarily change system settings
- Path to home can get complicated
  - `~` (twiddle or tilde) is shorthand for your home directory
  - `$: cd` will take you there

**Orienteering: Leaving Home**

- Where am I? `pwd`
  - Multiple accounts on same/different machines
- Who am I? `whoami`
  - Very easy to log into a string of machines
- Where am I? `hostname`
  - Easy to log into a string of machines
- Try it and learn a little about yourself!

**Info**

- So far we have identified where we are and looked at paths
- Finder lets you see what is in a location
- `ls` has many arguments
  - So do most UNIX commands

**Looking Around: List (ls)**

**Demo**

- If you are somewhere else and want to start from home, `~` helps you
- Instead of:
  - `$: cd Users/jdebarry/Documents`
  - Use:
  - `$: ~/Documents`

**Orienteering: Leaving Home**

- Making the most of `cd` (Demos below)
- Moves from place to place
  - `$: cd`
  - `$: cd Desktop`
  - Prompt changes! At home it is `~`
- `.` is another way to specify your current location
- `..` specifies one directory up
  - Both are hidden in whatever directory you are in (more in a moment)
- In lectures, I use `#` to make a comment, not part of the command
  - `pwd` #should be Desktop
  - `$: cd ..` #Moves you one directory up

- Can be relative or absolute
  - `$: cd ../../` #up 2 directories from any place, relative to start location
  - `$: cd ../User/Documents` #up one directory and then an absolute path

**Orienteering: Leaving Home**

- Where am I? `pwd`
- Who am I? `whoami`
  - Multiple accounts on same/different machines
- Where am I? `hostname`
  - Very easy to log into a string of machines
- Try it and learn a little about yourself!

**Demo**

- `$: cd` #home
  - `$: ls` #list contents
  - `$: ls -l` #list long
  - `$: ls -a` #list hidden (`.` `..`)
  - `$: ls -F` #fancy
  - `$: ls --` #combine options
  - `$: ls /` #list root
  - `$: cd ~/Desktop` #go to root
  - `$: cd ..` #check location
  - `$: ls ~` #list home contents
UNIX: Commands

• Info
  • Already using: `pwd`, `open`, `cd`, `ls`, `whoami`, `hostname`
  • Standalone, executable programs built into the shell
  • When you type a command, UNIX looks for it in a list of programs (more on that later) and executes it
    – If it is not in the list you get "Command Not Found"

• Manual (`man`)
  – View 'man page' with info on command
  – Arguments are at the top (of most)
  – `apropos` – Searches man pages for commands associated with a keyword
    – Could make lots of output
  – Whitespace…
    – In general, BAD
    – Avoid in filenames, directory names
    – In commands, separates components (`ls -l`)
    – If using whitespace in a command, contain it in quotes `'`

• Demo
  $: man ls # space = page, return = line, b = back, q = quit
  $: ls –aF
  $: man apropos
  $: apropos column
  $: mail -s 'A Test Message' 'Your Name <username@uga.edu>'
  # Then type body of message and hold down control and hit d (ctrl-d).
  # This will send you an email

The “man” command “man ls”

• Google, Wikipedia
  – Info for and about UNIX commands
  – If you have a problem, likely there are others who have too!
• Use the UNIX/LINUX “man” command for manual
  • Not using your Internet resources to find information on available commands is a common error

Command Line Editing

Info
• Prolonged typing is not good for you
• Type less and do more
• Tab to complete paths or commands
• Clear screen
• Page up and down scrolls through terminal history

Demo
$: cd
# Don’t hit enter after the next command!
$: ls ...
# ‘hit tab’ once, what happens?
$: cd ...
$: pwd
$: clear
$: cd hit return to go back home

Info
• history
  # shows command history, numbered
  $: !\# from_history
  # without quotes, from history list, executes that command number
  $: !!
  # reissues previous command
**File Types and Permissions**

Info
- File: text, binary, etc.
- Directory (folder), executable program
- Making files
  - Editor or Terminal
- Make folders
  - Finder or Terminal
- Careful naming!
  - No special characters or whitespace
- "Test" tell me it is for playing

Demo
- Open BBEdit
  - Type Name
  - Save to Desktop as Test_1
- $ cd 
  - $ cd Desktop 
  - $ cat > Test_2
  - $ type Date
  - Ctrl+c
  - From Finder: Shift+Command+N
  - $ mkdir Test_3
  - $ ls
  - ls #See what you made
  - $ ls -l #File vs Directory

- 10 slots: File type, 3X Owner (you), 3X Group (not you, but the same Campus, Class etc), 3X Other (Everyone else who is not owner or group)

**File Types and Permissions**

- r: read
  - File: allows that file to be read, provided the directory containing it has x permission.
  - Directory: allows that directory to be listed with filenames only.
- w: write
  - File: allows that file to be written (altered or replaced).
  - Directory: allows new files to be added to that directory and existing files to be removed.
- x: execute
  - File: makes that file executable (if possible, file can be run as a program).
  - Directory: allows a detailed listing (if r is also set), and allows files to be read or written in the directory according to their individual file permission settings.

**Reading Files**

- cat can read and combine files
- '>' specifies that the output of the command be written to a file
  - When cat was used to make a file and to combine a file
- Can't act on things that are not in the working directory. Unless you are in the same place, or give the file path