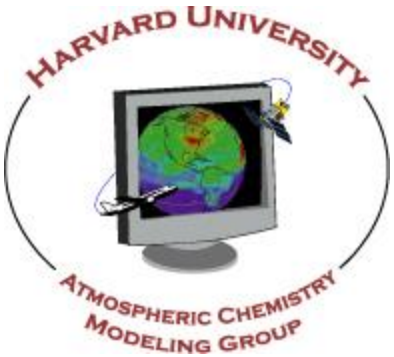
GEOS-Chem for Beginners

Melissa Sulprizio and Junwei Xu
GEOS-Chem Support Team
geos-chem-support@as.harvard.edu



The 7th International GEOS-Chem Meeting
04 May 2015

Overview

1) Getting started with GEOS-Chem
2) Downloading the shared data directories
3) Creating run directories
4) Compiling using makefile options
5) Debugging code
6) Visualizing output data
7) Q & A

We will focus on what is new in GEOS-Chem v10-01.

GEOS-Chem Documentation

- GEOS-Chem Website:
  http://www.geos-chem.org
- GEOS-Chem Public Wiki:
  http://wiki.geos-chem.org
- GEOS-Chem Users' Guide:
  http://manual.geos-chem.org
- GAMAP Users' Guide:
  http://acmg.seas.harvard.edu/gamap/doc/

Join GEOS-Chem Working Groups

All GEOS-Chem users are invited to join the Working Groups that corresponds to your own particular areas of research.

You are also encouraged to sign up for the relevant WG email list and add your project to the group's page on the GEOS-Chem wiki:

Adjoint Model and Data Assimilation	geos-chem-adjoint@seas.harvard.edu
Aerosols	geos-chem-aerosols@seas.harvard.edu
Carbon Cycle	geos-chem-carbon@seas.harvard.edu
Chemistry-Climate	geos-chem-climate@seas.harvard.edu
Hg and POPs	geos-chem-hg-pop@seas.harvard.edu
Organics	geos-chem-organics@seas.harvard.edu
Oxidants and Chemistry	geos-chem-oxidants@seas.harvard.edu
Nested Model	geos-chem-regional@seas.harvard.edu
Sources and Sinks	geos-chem-emissions@seas.harvard.edu
Transport	geos-chem-transport@seas.harvard.edu

Getting Started

- Make sure that you meet the minimum system requirements
- Download the GEOS-Chem source code using Git:

```
git clone git://git.as.harvard.edu/bmy/GEOS-Chem Code.v10-01
```

- Install the netCDF libraries:
  - Most input data files are now in COARDS-compliant netCDF format
  - Use `module load netcdf` to see if netCDF libraries are on your system
  - Install netCDF libraries using the `GEOS-Chem-Libraries` installer package
  - Link to the libraries from GEOS-Chem in your setup file (`.cshrc`, `.bashrc`).
  - Instructions are on the Installing libraries for GEOS-Chem wiki page.
- Download shared data directories
- Create GEOS-Chem run directories

Shared Data Directories

- The GEOS-Chem shared data directories contain input data read in during a simulation, such as:
  - Meteorological data
  - Emissions inventories
  - Other atmospheric data (e.g. prod/loss rates, oxidant fields)

- You may download the GEOS-Chem shared data directories from one of two archives using the wget utility:

  1. Harvard archive (ftp.as.harvard.edu)

     ```
     wget -r -nH ftp://ftp.as.harvard.edu/gcgrid/geos-chem/data/DIRNAME/
     ```

  2. Dalhousie archive (http://rain.ucis.dal.ca)

     ```
     wget -r -nH "ftp://rain.ucis.dal.ca/DIRNAME/"
     ```

ExtData Directory

- In GEOS-Chem v10-01, all data directories are now subdirectories of a single root directory called `ExtData`
- Code updates required a change in the data directory naming structure:

  `Root data directory : /dir/to/data/GEOS_4x5`

  is now

  `Root data directory : /dir/to/data/ExtData`

- `ExtData` contains the following data directories:
  - **`CHEM_INPUTS`**: non-emissions data for the GEOS-Chem chemistry modules
  - **`HEMCO`**: emission inventories and other data sets for use with HEMCO
  - Symbolic links to directories that store met fields (e.g. `GEOS_4x5`, `GEOS_2x2.5`)
- You or your sysadmin can add symbolic links from `ExtData` to the existing data directories if already downloaded

Meteorology Fields

- The Dalhousie archive stores met fields for the various nested grids that are not available on the Harvard archive.

Met field	Resolution	Availability
GEOS-FP	Global: 4° x 5°, 2° x 2.5° Nested: 0.5° x 0.625°, 0.25° x 0.3125°	2012/04/01 – 2015/03/31
GEOS-5	Global: 4° x 5°, 2° x 2.5° Nested: 0.5° x 0.666°	2003/12/01 – 2013/06/30
MERRA	Global: 4° x 5°, 2° x 2.5°	1979/08/01 – 2015/02/28
GEOS-4	Global: 4° x 5°, 2° x 2.5°	1985/01/01 – 2007/01/01

- Nested grids are available for China, Europe, North America (GEOS-5, GEOS-FP) and for SE Asia (GEOS-FP)

HEMCO Data Directories

- Emissions data are now stored in the HEMCO data directories
- HEMCO has online regridding capabilities, so we no longer need to store emissions data on multiple grids
- Data are stored in COARDS-compliant netCDF files and at the native resolution when possible
- Each folder of the HEMCO data directory tree represents a particular emission inventory or atmospheric data set
- Download the `hemco_data_download` package for assistance:

```
git clone https://github.com/GCST/hemco_data_download.git
```

  - Use this package to select the emission inventories to download
  - The `hemcoDataDownload.rc` input file is set up to download HEMCO data directories used in the GEOS-Chem benchmark simulations

GEOS-Chem Unit Tester

- The GEOS-Chem Unit Tester is a package that will compile and run several simulations with a standard set of debugging flags
- A unit test looks for parallelization, floating invalid, and array-out-of-bounds errors in each simulations for a given combination of:
  1. Meteorology field type
  2. Horizontal grid
  3. Simulation type
- Download the GEOS-Chem Unit Tester using Git:

```
git clone git://git.as.harvard.edu/bmy/GEOS-Chem-UnitTest
```

- The GEOS-Chem Unit Tester creates a web page summarizing the results of your unit test in an easy-to-read table
- For more information see the Debugging with the GEOS-Chem Unit Tester wiki page

Create Run Directories

- GEOS-Chem simulations are performed in a run directory
- Use a different run directory for each combination of met data, grid, and simulation type (e.g. `geosfp_4x5_fullchem`)
- Run directories for GEOS-Chem v10-01 are now created with the GEOS-Chem Unit Tester
  1. Navigate to the `GEOS-Chem-UnitTest/perl/` directory
  2. Edit `CopyRunDirs.input` to select the run directories that you wish to create
  3. Execute the `gcCopyRunDirs` perl script to create the run directories:

     ```
     gcCopyRunDirs CopyRunDirs.input
     ```

- Detailed instructions can be found on the Creating GEOS-Chem run directories wiki page

Setting Run Options

- Configurable files in a run directory include:
  - **`Makefile`**: compiles and runs GEOS-Chem
  - **`input.geos`**: sets run options
  - **`HEMCO_Config.rc`**: sets emission options
- Check the following settings in `Makefile`:
  - **`CODE_DIR`**: GEOS-Chem source code directory path
  - **`LOG_DIR`**: output log path
  - **`VERSION`**: prefix for output log filename
- Check the following settings in `input.geos`:
  - Start and end dates
  - Data directory file paths
  - Options for transport, convection, deposition, chemistry, etc.
  - Options for diagnostics and timeseries output

Setting Emission Options

- Emission options are now specified in the HEMCO configuration file, `HEMCO_Config.rc`
- Check the following settings in `HEMCO_Config.rc`:
  - **`ROOT`**: path to the HEMCO data directories
  - **`Logfile`**: name of HEMCO log file, default is `HEMCO.log`
  - **`DiagnPrefix`**, **`DiagnFreq`**: HEMCO emission diagnostics settings
    - Frequency options are `Hourly`, `Daily`, `Monthly`, `Annual`, `End`, or `Manual`
    - The emission diagnostics will be saved out to either the GEOS-Chem bpch file or the HEMCO diagnostic file, but not both
  - **`Verbose`**, **`Warnings`**: range from 0 (no log output) to 3 (a lot of log output)
  - Base section extension switches that turn emissions on or off
    - Use `on`/`off` to set emission extensions (e.g. Base, PARANOX, MEGAN, GFED, lightning NOx)
    - Use `true`/`false` to set emission inventories

Compiling GEOS-Chem

- You can compile and run your GEOS-Chem simulation from your run directory using `Makefile`
- Some important compiling commands:
  - **`make help`** : shows help screen with all possible make options
  - **`make -j4 ...`**: compiles 4 files simultaneously (you can pick a different #)
  - **`make -j4 all`**: runs a GEOS-Chem unit test
  - **`make -j4 mp`**: compiles & runs GEOS-Chem with parallelization on
  - **`make -j4 mp TRACEBACK=yes`**: compiles & runs with traceback turned on
  - **`make -j4 mp BOUNDS=yes`**: compiles & runs w/ array-out-of-bounds checks
  - **`make -j4 mp FPE=yes`**: compiles & runs w/ floating-point exceptions checks
  - **`make fileclean`**: removes all output files in the run directory
  - **`make realclean`**: removes all compiled files in the code directory
  - **`make superclean`**: runs `make fileclean realclean`

Debugging Tips

- Check the GEOS-Chem log file and the HEMCO log file
  - Turn on `ND70` in `input.geos` to print debug output to the log file
  - Set `Verbose` to 3 in `HEMCO_Config.rc`
- Compile with debug options turned on:
  - **`BOUNDS=yes`** turns on array out-of-bounds error checking
  - **`TRACEBACK=yes`** returns a list of routines that were called when the error occurred
  - **`FPE=yes`** checks for floating-point exceptions (div-by-zero, NaN, etc.)
  - **`DEBUG=yes`** compiles GEOS-Chem for use in a debugger (Totalview, IDB)
- Isolate the error to a particular operation by turning on/off transport, chemistry, drydep, etc.

Please report any bugs in the standard code to the GC Support Team and post on the GC wiki for other users to see!

Visualizing Output with GAMAP

- GEOS-Chem diagnostic output is saved out in binary punch format
- Use ***GAMAP*** (a software package written in IDL) for GEOS-Chem output visualization tasks
  - Easy-to-use interactive interface fulfills basic plotting needs
  - See http://acmg.seas.harvard.edu/gamap for more info
- GAMAP subroutines can be used as standalones in IDL. For example,
  - **`CTM_GET_DATA`***:* gets several data blocks
  - **`CTM_PLOT`**: reads data from disk and plots it
  - **`CTM_SUM_EMISSIONS`**: sums up emissions totals
  - **`BPCH2COARDS`**: converts bpch file to COARDS-compliant netCDF file
- Several GEOS-Chem users are also developing packages in Python for use with GEOS-Chem (e.g. PyGChem, bpchdump.py, Gchem)

Good Coding Practice

- Use copious comments
  - Comments will make sure that current and future GEOS-Chem users will be able to understand the modifications that you have added
- Use ProTeX headers
  - ProTeX headers are specially formatted comments at the top of each subroutine, function, and module that list arguments, modification history, and references
  - Documentation is automatically generated when you compile with `make doc`
- Follow these tips for writing effective Fortran code
  - Indent new blocks of code within DO loops and IF statements
  - Use lots of white space to separate code
  - Use Fortran-90 instead of Fortran-77
  - Structure DO loops efficiently (loop over arrays using `N`, `L`, `J`, `I` order)

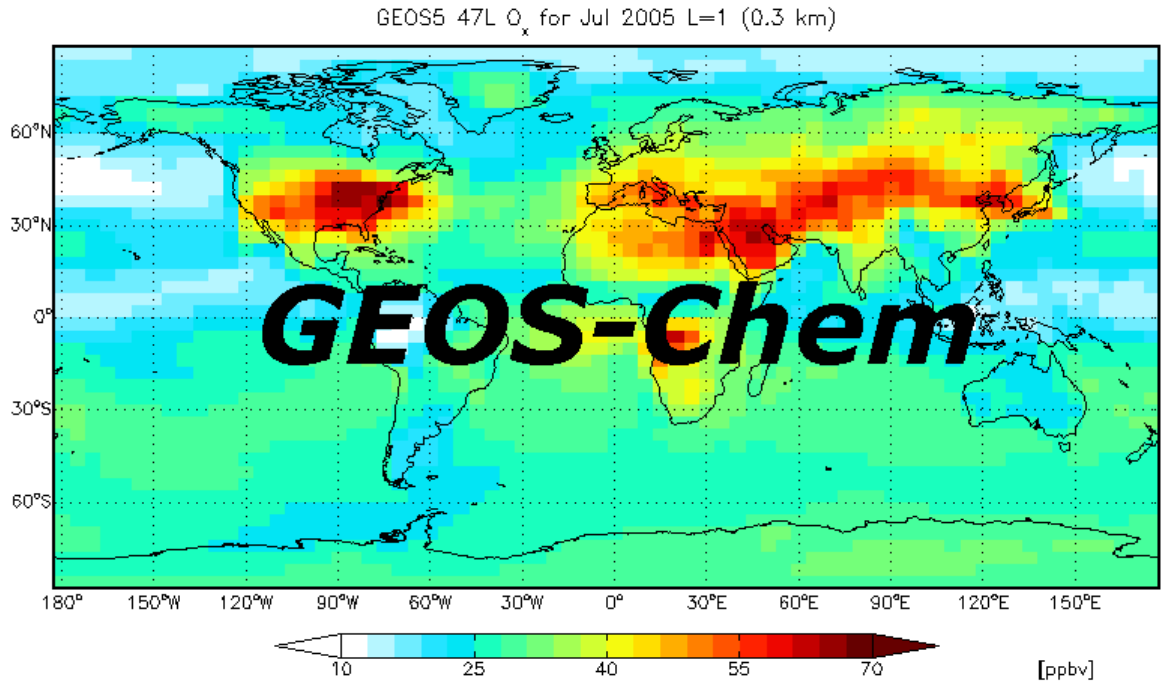
Submitting Code Updates

- Test your code with the GEOS-Chem Unit Tester to make sure your simulation works for other met fields, resolutions, and simulation types
  - Fix any parallelization, floating invalid, or array-out-of-bounds errors
  - Remove any array temporaries that are created
- Contact your Working Group leaders and the GEOS-Chem Support Team to request that your changes be included in the standard code
- When submitting updates please include the following:
  1. Source code updates in a Git patch file
  2. Data files in COARDS-compliant netCDF format
  3. Documentation (e.g. README files, a wiki post)

Q & A

Would you like to go over any specific topics?

Would you like to see any GEOS-Chem demos?



Extra Slides

Git Version Control Software

- Git version control software is a distributed source code management system
- Offers many improvements over previous source code management software (e.g. CVS, Subversion)
  - Allows you to keep an identical copy of the GEOS-Chem source code repository on your own system
  - Lets you easily combine code from several developers
  - Contains easy-to-use graphical tools
- Git must be installed on your system
  - Git is free, open source software
  - You or your sysadmin can obtain the Git source code from the Git website
- GEOS-Chem source code, GEOS-Chem Unit Tester, and GAMAP visualization software are distributed with Git

Downloading and Updating

`git clone`

- Gives you a clean copy of the “master” repository
- Provides source code files + total version history
- Use this command to download a new version of the code

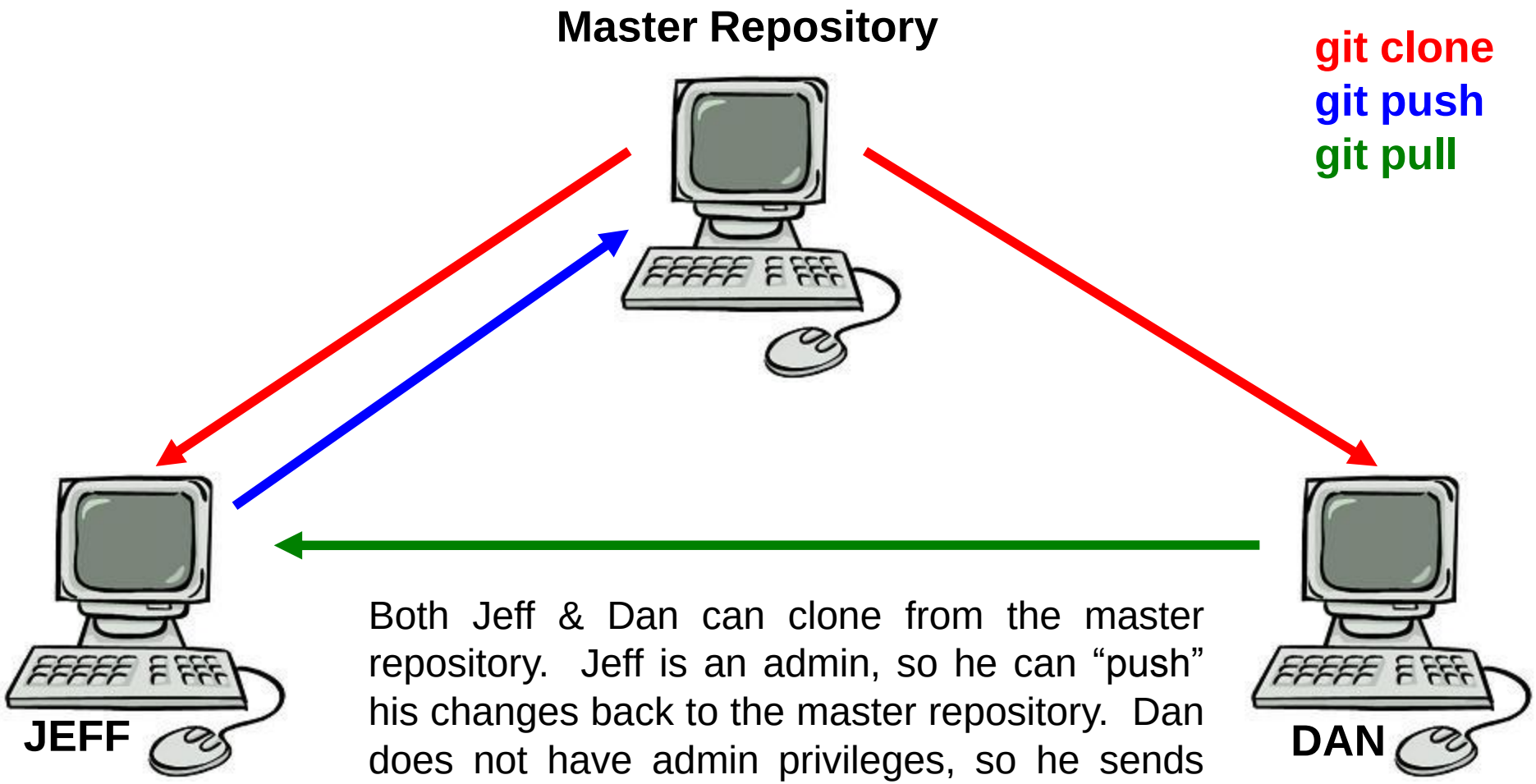
`git pull`

- Gets updates from the “master” repository and puts them into your existing local repository
- Use this command to apply bug fixes (aka “patches”) to your code or to get code from another user's local repository

`git push`

- Uploads code from your local repository to the “master” repository
- Typically this command will only be used by people with admin privileges

Downloading and Updating



Both Jeff & Dan can clone from the master repository. Jeff is an admin, so he can “push” his changes back to the master repository. Dan does not have admin privileges, so he sends his updates to Jeff, who then “pushes” them to the master repository.

Graphical Tools

`gitk`

- This command opens the GitK window where you can browse the complete revision history
- For each commit, you can see the changes made to each file

`git gui`

- This command opens the Git GUI, a graphical user interface that can be used for most Git commands
  - Create new branches of development
  - Commit (and document) changes to the repository
  - Merge branches back into the “master” line of development

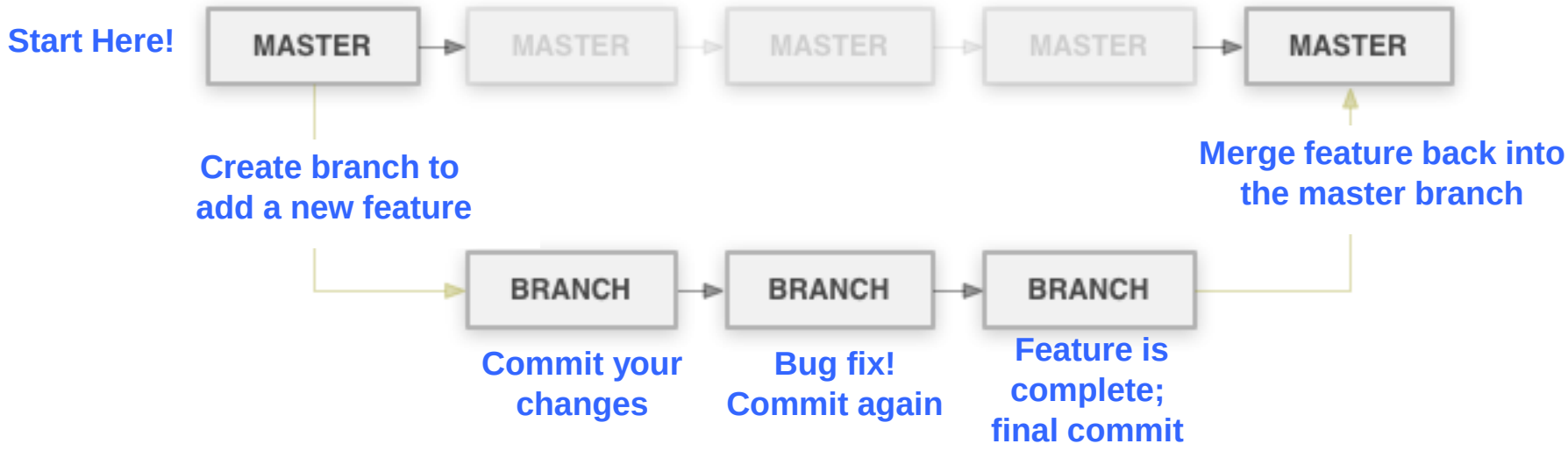
NOTE: We recommend using the Git GUI rather than typing commands at the Unix prompt. The git gui greatly simplifies things.

Branching Commands

- Git allows you to split model development from the main line of development (the “master” branch) into one or more branches
- We recommend that you create a new branch in which to make code changes, rather than making updates in the “master” branch
- Creating a branch
  - In git gui: Branch / Create
  - The new branch is checked out after it is created
- Checking out a branch
  - In git gui: Branch / Checkout
  - The checkout command switches the files in your source code directory so that they correspond to the branch you requested
  - You should always commit your updates to the current branch before checking out a new branch

Branching Commands

- Merging branches together
  - In git gui: Merge / Local Merge
  - This will merge code from the branch that you pick into the current branch
- Deleting branches
  - In git gui: Branch / Delete
  - Once you have merged a branch you can delete it



Reference: http://hoth.entp.com/output/git_for_designers.html

Committing Changes

Committing is best done in the Git GUI

- ***Unstaged changes*** (top-left GUI window): Files w/ updates that Git does not know about yet. Click on them to stage.
- ***Staged changes*** (bottom-left GUI window): Files that will be added to the local repository when you commit.
- ***Commit message*** (lower-right GUI window): You can type a message describing the files/directories that are being committed. The first line of the message will show up as the commit title in the GitK browser.
- ***Sign-off button*** (lower-right GUI window): Click this button to add your name & email to the commit message.
- ***Commit button*** (lower-right GUI window): Click this button to commit your updated files/directories to the repository.
- ***Amend Last Commit*** (lower-right GUI window): Check this box if you want to update the message from the last commit.

Sharing Code with Others

`git pull`

- This command can be used to “pull” code from another user’s repository
- This can only be done when you can “see” the other repository on disk or via the internet

Patch file

- Text file containing the changes between commits that can be sent to another user via email
- A patch file can be created with this command :

```
git format-patch -1 --stdout > my_patch_file.txt
```

- A patch file can be ingested into your local repository with this command:

```
git am < my_patch_file.txt
```

- For a patch file to work, you have to know its **parent commit** (i.e. the commit just prior to the point where the patch file was made)