Anleitung zum Arbeiten mit der Marburg Software Library

Philipp Keding, *keding@mathematik.uni-marburg.de*

Version 1.0, Oktober 2014

Die *Marburg Software Library* (kurz: *MSL*) ist template-basierter c++-Quellcode zum Lösen von partiellen Differentialgleichungen auf Gebieten. Sie wurde von den damaligen Doktoranden der AG Numerik Thorsten Raasch und Manuel Werner ins Leben gerufen und seither von vielen Doktoranden der Arbeitsgruppe zum Zwecke numerischer Experimente verwendet und weiterentwickelt. Mittlerweile befindet sich die Software als Open Source online auf dem Github-Server (`www.github.com`). In dieser Anleitung wird beschrieben, wie man unter Verwendung des Versionsverwaltungssystems *Git* Zugriff auf die *Marburg Software Library* bekommt, um an ihr zu arbeiten und Änderungen an ihr durchzuführen. Hierbei wird davon ausgegangen, dass Ubuntu oder eine andere Linux-Distribution verwendet wird. In diesem Tutorial erklären wir das Verwalten des Projekts über die Kommandozeile. Es sei aber darauf hingewiesen, dass auch andere Programme wie z. B. Netbeans Schnittstellen zu Git haben, welche eine komfortablere Programmierumgebung bieten. Ein sehr gutes Online-Tutorial zu Git findet man hier: `https://git-scm.com/book/de/v1/`.

1 Installation und Konfiguration von Git

Git ist ein sogenanntes distribuiertes Versionsverwaltungssystem. Dies erlaubt es dem Benutzer zunächst auf einem lokalen Repository (Aufbewahrungsort) Änderungen der Software durchzuführen und zu experimentieren, bevor er sie in das externe Repository (befindet sich auf Github) einspeist. Zum installieren verwendet man den Befehl

```
$ sudo apt-get install git
```

in der Kommandozeile.
Nach der Installation solltest du mit den Befehlen

```
$ git config --global user.name "[NAME]"
$ git config --global user.email [EMAIL-ADRESSE]
```

deinen Namen und deine E-Mail Adresse konfigurieren. Weitere nützliche Konfigurationen sind

```
$ git config --global core.editor [EDITOR]
$ git config --global merge.tool [DIFF-PROGRAMM]
```

zur Auswahl eines Editors (z.B.: nano) und eines Diff-Programms (z.B.: vimdiff), welches später benötigt wird, um Konflikte zu lösen, die während der Arbeit mit Git auftauchen können. Zum Überprüfen der Einstellungen verwende

```
$ git config --list
```

Bemerkung Um Hilfe in der Anwendung von Git zu erhalten, können die Befehle

```
$ git help <verb>
$ git <verb> --help
$ man git-<verb>
```

nützlich sein, wobei man `<verb>` durch den Ausdruck, zu dem man Hilfe benötigt, ersetzt (z.B.: config).

2 Einbinden der Marburg Software Library

Die MSL befindet sich auf dem Github Server. Um darauf Zugriff zu erlangen, wechselt man in ein Verzeichnis seiner Wahl (z.B.:Desktop) und führt den Befehl

```
$ git clone https://github.com/agnumerikunimarburg/Marburg_Software_Library.git
```

aus. Dadurch wird ein Verzeichnis mit dem Namen `Marburg_Software_Library` erstellt, in das der gesamte Inhalt des Repositorys auf dem Server hineinkopiert wird. Durch den clone-Befehl wurde die Quelle automatisch als Remote Repository (extern) mit dem Namen `origin` abgespeichert. Um die Installation der MSL abzuschließen, muss man lediglich noch die ënvironment variablesff- ür die einzelnen Verzeichnisse richtig setzen. Hat man das Repository direkt in den Homeordner runtergeladen, trägt man dazu

```
export MATHTL_DIR=$HOME/Marburg_Software_Library/MathTL
export WAVELETTL_DIR=$HOME/Marburg_Software_Library/WaveletTL
export FRAMETL_DIR=$HOME/Marburg_Software_Library/FrameTL
```

in eine beliebige freie Zeile der Datei `.bashrc` (öffnen z.B. mit dem Befehl `$ gedit ~/.bashrc`) ein. Will man in Zukunft seine Daten nun lokal mit Daten des Remote Repository zusammenführen (mergen), genügt der Befehl

```
$ git pull origin
```

3 Arbeiten im lokalen Repository

Git selbst unterscheidet zwischen drei Bereichen: Arbeitsverzeichnis, Staging Area und Repository.

- Das Arbeitsverzeichnis enthält den aktuellen Stand des Projektes und alle noch nicht erfassten Änderungen
- Die Staging Area beinhaltet alle Änderungen, die bei einem Commit (=Einbringen einer Aktualisierung in das Repository) einer neuen Version dem Repository hinzugefügt werden.
- Das Repository beinhaltet alle Versionen des Projektes. Diese werden samt aller Verwaltungsinformationen für das Repository im Ordner .git gespeichert.

Mit

```
$ git status
```

bekommt man angezeigt, welche Dateien seit dem letzten Commit neu dazugekommen sind oder modifiziert wurden. Möchte man diese nun dem lokalen Repository hinzufügen, stellt man sie zunächst mit

```
$ git add [DATEI]
```

in die Staging Area (**Wichtig !!! Nur .cpp und .h Dateien ins Repository hinzufügen**). Möchte man die Dateien in der Staging Area nun dem lokalen Repository hinzufügen, geht das mit dem Befehl

```
$ git commit
```

Daraufhin öffnet sich dre Editor. Hier sollte man den Commit so kommentieren, dass nachvollziehbar wird, was geändert wurde. Es bietet sich daher auch an, Änderungen, die zusammengehören, in Commits zu gruppieren und nicht direkt alle Änderungen zu comitten.

4 Änderungen auf externes Repositiory hochladen

Möchte man Änderungen auf das externe Repository packen, benötigt man zunächst einen Account auf `www.github.com`. Nachdem man sich angemeldet hat, bitte eine E-Mail an mich (keding@mathematik.uni-marburg.de), mit der Bitte, in die Liste der Mitwirkenden aufgenommen zu werden. Bevor man nun auf das externe Repository hochläd, sollte man zunächst mit

```
$ git pull origin
```

lokal die Repositorys zusammenführen (falls jemand anderes auf das externe Repository zuvor hochgeladen hat). Gibt es hier Konflikte zwischen Dateiversionen müssen diese erst lokal gelöst werden (helfen kann hier ein Diff Programm). Danach aktualisiert man das lokale Repository. Ist es nun in dem Zustand, dass es hochgeladen werden kann (Bitte erst lokal schauen, ob auch alles kompiliert), macht man das mit dem Befehl

```
$ git push origin master
```

und darauffolgender Eingabe von Benutzername und Passwort auf Github. Hier wurde nur ein kleiner Ausschnitt gezeigt von dem, was Git alles kann. Insbesondere wurde nicht auf eine der wohl nützlichsten Eigenschaften von Git, das Arbeiten mit Branches (Zweigen), eingegangen. Hier sei nochmal auf das Tutorial am Anfang hingewiesen.