Institut für Physikalische Chemie

Übungen zum Methodenkurs „Programmierkonzepte in der Physikalischen Chemie“ im WS 2013/2014

Dr. Till Biskup

— Lösungen zum Aufgabenblatt 4 vom 22.01.2014 —

Eine wichtige Anmerkung vorweg: Die Fragen auf dem zugehörigen Aufgabenblatt entziehen sich aufgrund ihres Wesens einer „Musterlösung“ – noch weitaus mehr, als das für den Quelltext einer Programmieraufgabe gilt. Alle nachfolgenden Ausführungen sind deshalb notwendigerweise stark subjektiv und stellen nur eine mögliche Antwort dar, die zwar auf langjähriger persönlicher Erfahrung beruht, aber daraus keinerlei Autorität ableiten kann.

Aufgabe 4—1

Die Lektüre des Git-Buches und die Installation des Programmes kann nur in Eigenregie erfolgen. An dieser Stelle nur noch einmal die Ermutigung: Spielen Sie mit dem System, probieren Sie es einfach bei nächster sich bietender Gelegenheit mit einem eigenen (realen) Projekt aus. Meine Erfahrung: Ich möchte die Vorzüge von Git nicht mehr missen.[1]

Ein entscheidender Vorteil, den Git mit anderen dezentralen (verteilten) Versionsverwaltungssystemen teilt und der diese Systeme von serverbasierten Lösungen wie CVS oder Subversion (und ihren kommerziellen Derivaten) unterscheidet, ist die Unabhängigkeit von anderen Rechnern (insbesondere Servern). Sie können auf Ihrem Rechner mit Ihrem Git-Repository arbeiten, ganz unabhängig von einer bestehenden Verbindung zum Internet. Das ist gerade dann nützlich, wenn man mobil unterwegs sein möchte.[2] Ganz nebenbei ist die gesamte Historie immer lokal verfügbar, weshalb auch komplexe Abfragen und Operationen schnell und unkompliziert möglich sind.

Beim verteilten Arbeiten an gemeinsamen Projekten sollten Sie es sich zur Regel machen, zumindest jedesmal, wenn Sie wieder beginnen, am Projekt zu arbeiten und Dateien zu ändern, nach Änderungen durch Ihre Mitstreiter zu schauen. Das wird in aller Regel über ein `pull` von einem entfernten Repository zu bewerkstelligen sein.

Darüber hinaus ist es gute Praxis, jeweils am Ende der aktiven Arbeit am Projekt, spätestens jedoch am Ende eines jeden Tages, die bisherigen Änderungen mit einem Commit in das Repository zu übertragen und vor allem diese Änderungen auch via `push` den anderen Mitstreitern zugänglich zu machen. Sollte es nicht möglich sein, funktionsfähige Versionen zur Verfügung zu stellen, bieten sich entsprechende Zweige (*branches*) an. All das kann und soll natürlich nicht die Kommunikation der Projektpartner untereinander ersetzen, erleichtert aber mitunter in der Praxis die Zusammenarbeit.

[1] Ironie am Rande: Es ist auch nur etwa zwei Wochen her, dass ich sämtliche (LaTeX-)Quelltexte sowohl für die Folien als auch für die Übungsaufgaben (inklusive dieser) und zugehörige Programmbeispiele in ein Git-Repository einspeiste. Manchmal ist das Offensichtliche eben doch zu nah…

[2] Wer einmal mit dem Zug quer durch Deutschland gefahren ist und dabei versucht hat, über das Internet zu arbeiten, weiß diesen Aspekt der Netzunabhängigkeit zu schätzen.

Aufgabe 4—2

Ein für den Entwickler brauchbarer (und in diesem Kontext „sinnvoller“) Bug-Report (Fehlerbericht) sollte nach Möglichkeit alle für die Behebung notwendigen Informationen beinhalten. Das sind, grob zusammengefasst, eine kurze Beschreibung, in welchem Kontext der Fehler auftrat, die genaue Fehlerausgabe des Programms oder der Funktion, Eckdaten zum Betriebssystem (Name, Version, ggf. Service Pack, Architektur – 32 oder 64 bit) und zum verwendeten Programm (z.B. bei Matlab die Matlab-Version) sowie ggf. die genaue Version der Toolbox. Letzteres setzt ein eineindeutiges Schema für Versionsnummern[3] sowie einen einzelnen Ort für deren Pflege[4] voraus.

Wenn Sie sich überlegen, im Rahmen einer eigenen Toolbox eine Hilfsroutine zu schreiben, die möglichst viele Informationen über die Toolbox und das verwendete System bereitstellt, die dann im Fehlerfall gemeinsam mit dem Bug-Report an den Entwickler weitergegeben werden können, dann stellt Ihnen Matlab eine Reihe an Möglichkeiten bereit, automatisiert einen Großteil dieser Informationen zu erhalten. Über Funktionen wie `computer` erhalten Sie Informationen über das verwendete Betriebssystem.[5] Informationen zur installierten Matlab-Version liefert der Befehl `ver`, und wenn er mit einem Rückgabeparameter aufgerufen wird, dann lassen sich die Informationen auch noch weiterverarbeiten. Die bereits in der Fußnote weiter oben erwähnte Datei `Contents.m` ist eine sehr gute Möglichkeit, wichtige Informationen zu Ihrer Toolbox wie die Versionsnummer abzulegen.

Den aktuellen Nutzer[6] brauchen Sie für einen Fehlerbericht nicht – diese Information wäre im Sinne einer Datensparsamkeit auch gar nicht erwünscht. Gleiches gilt für die Daten, die zum Zeitpunkt des Auftretens des Fehlers verarbeitet wurden. Geben Sie *niemals* ungefragt die Daten weiter und kommunizieren Sie den Nutzern Ihrer eigenen Programme, das ebenfalls nie unaufgefordert zu tun.

Was schließlich noch fehlt, ist die genaue Fehlerausgabe. Matlab stellt Ihnen hier über seinen Fehlerbehandlungsmechanismus, in dessen Zentrum die Klasse `MException` steht, und Befehle wie `dbstack` Werkzeuge zur Verfügung, diese Informationen detailliert auszuwerten und automatisiert zu einem Fehlerbericht hinzuzufügen.

Aufgabe 4—3

Die Einteilung einer Dokumentation nach Zielgruppen kann z.B. entlang der großen Gruppen „Entwickler“ und „Anwender“ erfolgen. Auch wenn oft die Entwickler auch Anwender sind und zumindest zuweilen Anwender zu Entwicklern werden, sind das doch in der Regel unterscheidbare Rollen mit unterschiedlichen Ansprüchen und Bedürfnissen.

Ausgehend von dieser Unterscheidung ist in groben Linien relativ klar, welche Information wo (bzw. wo nicht) abgelegt wird. Dokumentation im Quellcode selbst, die nach außen nicht sichtbar ist, kann sich nur an Entwickler richten. Die Schnittstellendokumentation, die z.B von Matlab über die Befehle `help` oder `doc` auch dem Anwender zugänglich ist, richtet sich vermutlich primär an Anwender, ist aber für Entwickler genauso von Belang.

---

[3] Versionsnummern einer Toolbox sind mit hoher Sicherheit unabhängig von einer Versionsverwaltung. Ein Beispiel für ein Schema für Versionsnummern ist „Semantic Versioning“, für Details vgl. `http://semver.org/`.

[4] Nur wenn die Versionsnummer einer Toolbox zentral an einer Stelle abgelegt wird, kann sichergestellt werden, dass zu jeder Zeit zumindest konsistente Informationen diesbezüglich verfügbar sind. Ein Beispiel, wie sich so etwas (im Kontext von Matlab) realisieren lässt, werden Sie in der zugehörigen Lektion zu Toolboxen (Lektion 8) sehen. Neugierigen sei der Blick in die Matlab-Hilfe zum Thema „Toolboxen“ empfohlen, Stichwort „`Contents.m`“.

[5] Genaueres finden Sie in der Matlab-Hilfe zum Befehl, ggf. müssen Sie sich über die Verweise auf andere verwandte Befehle „durchhangeln“, bis Sie das Gewünschte finden.

[6] Für Interessierte: An diese Information heranzukommen, ist in Matlab nicht zwangsweise offensichtlich oder trivial. Die Funktion `getenv` liefert abhängig vom Betriebssystem die gewünschte Information – die z.B. für eine Historie der Datenverarbeitung durchaus wieder ihre Berechtigung hat. Dazu in Lektion 6 mehr.

Alle konzeptionelle Dokumentation sollte nach Möglichkeit außerhalb des Quellcodes abgelegt werden. Welche Form der Dokumentation man hier wählt, ist wohl in weiten Teilen der persönlichen Vorliebe, den zur Verfügung stehenden Möglichkeiten und den eigenen Fähigkeiten geschuldet. Aus eigener Erfahrung kann ich sagen, dass ein Wiki aufgrund seiner einfachen Änderbarkeit bei gleichzeitiger Möglichkeit einer sinnvollen Gestaltung eine Möglichkeit ist, einen Großteil der Dokumentation aller Arten und für alle Zielgruppen zusammenzuführen.[7]

Ebenfalls aus eigener Erfahrung sei noch ein Tipp beigesteuert. Es ist mitunter erstaunlich hilfreich, im Hauptverzeichnis einer Sammlung von Funktionen[8] ein oder zwei Textdateien abzulegen, und zwar eine Datei mit dem Namen `README` und ggf. eine `INSTALL` betitelte Datei. Wie der jeweilige Name schon sagt, findet der geneigte Nutzer in der Datei `README` ein paar allgemeine Hinweise, was er eigentlich vor sich hat und was der oder die Entwickler sich ursprünglich einmal dachten, sowie idealerweise ein paar erste Hinweise zur Benutzung, die Datei `INSTALL` wird, wenn nötig, entsprechende Hinweise zur Installation zur Verfügung stellen.

Zum Abschluss: Wie könnte man sinnvoll dem Problem begegnen, dass Dokumentation oft hoffnungslos veraltet ist? Welche Hilfsmittel wären denkbar? Welche Kriterien müssten diese Hilfsmittel erfüllen?

Hierauf eine Antwort zu geben hieße, die Intention der Frage *ad absurdum* zu führen, ging es mir doch mit dieser Frage darum, selbst etwas zu lernen. Deshalb sei hier (noch) einmal explizit gesagt: (nach Möglichkeit) konstruktive Kritik und Anregungen aller Art sind herzlich willkommen.

[7] Es ist technisch durchaus ihm Rahmen der Möglichkeiten, die Schnittstellendokumentation automatisch in Wiki-Syntax zu übertragen und genauso automatisiert in ein Wiki zu integrieren.

[8] Die Unterscheidung zwischen einer Sammlung von Funktionen und Skripten und einer Toolbox sei hier nicht von Belang. Details dazu folgen in der entsprechenden Lektion.