Kurzeinführung[1]

1 Eine Sitzung am Rechner

Die Rechner im CIP-Pool (Raum 242 im Hauptgebäude) laufen mit dem Betriebssystem LINUX.

1.1 Das Einloggen (login)

1. Monitor einschalten, der Computer sollte durchgehend eingeschaltet sein (damit das Anmelden über das Netz möglich ist).
2. Geben Sie als Benutzernamen Ihre Matrikelnummer ein und danach Ihr Passwort.
3. War das Passwort korrekt, so wird eine grafische Benutzeroberfläche von LINUX gestartet. Und schon kann es losgehen.

1.2 Das Ausloggen (logout)

Wichtig: Wenn Sie Ihre Sitzung beendet haben, dürfen Sie nicht das Betriebssystem herunterfahren oder gar den Rechner abschalten, da ansonsten die Arbeit anderer Benutzer auf Ihrem Rechner empfindlich gestört wird!

1. Beenden Sie die grafische Benutzeroberfläche, indem Sie im Menü (normalerweise am unteren Bildschirmrand) entweder den Menüpunkt *Abmelden* auswählen oder den entsprechenden Knopf drücken. Wenn das Login-Fenster wieder erscheint, sind Sie ausgeloggt.
2. Monitor ausschalten.

1.3 Sicherheit und Passwörter

Vor dem ersten Anmelden, oder falls Sie ihr Passwort vergessen haben, können Sie unter `https://www.igpm.rwth-aachen.de/sicher/` ein neues Passwort festlegen.

Informationen zur Sicherheit von Passwörtern und wie man einfach merkbare und trotzdem sichere Passwörter erstellen kann, finden Sie zum Beispiel in [1].

[1] Stand: 11. April 2018

1.4 Aufgabenblätter und Dateien

Die Aufgaben und Zusatzinformationen zum Mathematischen Praktikum liegen auch online vor und sind im Internet unter

`https://www.igpm.rwth-aachen.de/mapra`

zu finden.

Auf den Rechnern im CIP-Pool können verschiedene Webbrowser, z.B. `firefox`, `google-chrome` und `opera` benutzt werden.

2 Rechnernutzung von zu Hause per Internet

Von zu Hause aus sind die Rechner

`mars.mathepool.rwth-aachen.de` und `venus.mathepool.rwth-aachen.de`

über das Internet zu erreichen. Sollte es Schwierigkeiten bei der Erreichbarkeit der Rechner geben, wenden Sie sich bitte per Email an das Team unserer Administratoren unter der Adresse `admin@igpm.rwth-aachen.de`. Die Administratoren werden versuchen, sich sobald wie möglich um eine Lösung des Problems zu bemühen.

Für den Zugriff auf die Rechner gibt es zwei Möglichkeiten:

2.1 Zugang per X2Go

Auf den Rechnern `mars` und `venus` läuft ein X2Go-Server. Mit dieser Software kann ein grafisches Benutzerinterface über das Internet, durch Datenkompression und Caching sogar über schmalbandige Leitungen übertragen werden. Um diesen Dienst nutzen zu können, benötigen Sie einen X2Go-Client. Dieser steht für alle gängigen Betriebssysteme auf der Webseite `http://wiki.x2go.org/` kostenlos zur Verfügung. Eine Dokumentation zur Konfiguration des X2Go-Clienten findet man unter `http://wiki.x2go.org/doku.php/doc:usage:x2goclient`.

2.2 Zugang per ssh und scp

Auf dem Rechner läuft außerdem ein ssh-Server. Über diesen kann mittels `ssh` eine Shell übertragen werden und mit `scp` können Dateien vom oder auf den Server kopiert werden. Die Software ist in den meisten Linux-Distributionen vorinstalliert. Ein geeigneter Client für Windows liegt zum Beispiel auf dem `ftp`-Server des Deutschen Forschungsnetzes (DFN) unter der Adresse `ftp://ftp.cert.dfn.de/pub/tools/net/ssh/` bereit.

Um sich mittels `ssh` auf einem der zur Verfügung stehenden Rechner einzuloggen, reicht es folgenden Befehl in der Konsole einzugeben:

```
ssh Benutzername@Host
```

Danach kommt nur noch die obligatorische Passwortabfrage und man ist auf „Host“ eingeloggt. Konkret heißt das, falls man sich von außen einloggen will:

```
ssh Benutzername@mars.mathepool.rwth-aachen.de   bzw.
ssh Benutzername@venus.mathepool.rwth-aachen.de
```

Informationen zu anderen Möglichkeiten, die `ssh` bietet, bekommt man indem man in der Konsole das Kommando `man ssh` (man für manual=Anleitung) eingibt.

3 Übungen zu Hause bearbeiten

Das Praktikum ist auf das Programmieren mit Linux ausgelegt. Der große Vorteil ist, dass Linux ein freies Betriebssystem ist und sämtliche zum Programmieren benötigte Werkzeuge frei und ohne viel Aufwand verfügbar sind.

Viele der Übungen benutzen trotzdem keine betriebssystemspezifischen Besonderheiten. Daher ist es im Prinzip möglich, einige Vorarbeit am Rechner zu Hause zu leisten, der nicht unter Linux läuft. Häufig gibt es jedoch Testroutinen, die die von Ihnen ermittelten Ergebnisse testen. Diese sind vorkompiliert und laufen daher nur unter dem jeweiligen Betriebssystem.

- Wer unter Linux arbeitet, kann zu Hause arbeiten und braucht sich nur die Übungen im Rechnerraum testieren zu lassen. Die Fachschaft Mathe, Physik, Informatik bietet auch jedes Semester eine Linux-Install-Party an, auf der man mit Unterstützung von Experten eine Linux-Distribution auf seinem Laptop installieren kann. Debian[2] ist eine Linux-Distribution, die man auch ganz einfach und schnell selbst installieren kann. Ist die Distribution installiert, kann man sofort mit dem Programmieren anfangen, da der GNU C++-Compiler[3], der auf den Praktikumsrechnern verwenden wird, vorinstalliert ist. Bitte beachtet dabei die Hinweise zur Compiler-Version in der C++-Kurzeinführung.
- Wer unter Windows arbeiten möchte, ist auf eine sogenannte Entwicklungsumgebung angewiesen, die eine vollständige Plattform zum Programmieren, Projektmanagement und Debugging bieten. Mit diesen Umgebungen kommen auch spezielle Compiler. Die folgenden Entwicklungsumgebungen bieten C++-Compiler: das Microsoft Visual Studio[4] oder der Intel Composer[5] (ICC). Man beachte, dass diese Entwicklungsumgebungen Lizenzen kosten. Eventuell besteht auch die Möglichkeit, eine Lizenz eines dieser Produkte über das IT Center der RWTH zu erhalten. Bitte wenden Sie sich dazu an das IT Center[6]. Für kosten- und lizenzfreie Compiler kann man z.B. MinGW (Minimal GNU for Windows[7]) installieren, um danach den GNU C++-Compiler installieren zu können. Das ist aber nur Leuten zu empfehlen, die wirklich Ahnung von ihrem Computer haben.

  Mit diesen Programmen kann der Code aber nur *im Prinzip* getestet werden, da die Testroutinen nicht aufgerufen werden können. Für die reine C++-Programmierung sind diese Tools natürlich geeignet und ausreichend.
- Wer mit Mac OS arbeiten möchte, kann auch den GNU C++-Compiler verwenden, da dieser auch in einer auf Mac OS funktionierenden Version existiert. Dennoch können die Testroutinen auch damit nicht aufgerufen werden. Alternativ ist der clang-Compiler[8] unter Mac OS recht verbreitet. Dieser steht unter einer BSD-ähnlichen freien Lizenz. Auch als Linux-Benutzer lohnt es sich mal clang anzusehen, da dieser oft deutlich verständlichere Fehlermeldungen generiert als gcc.

Will man die Testroutinen unter Mac OS oder Windows auch von Zuhause aufrufen, empfiehlt es sich, über Mars und Venus zu arbeiten (per X2Go oder `ssh`, siehe oben).

Wichtig: Die Abgabe der Aufgabe sollte jeweils möglichst nicht erst am letzten Termin erfolgen, da es sonst erfahrungsgemäß zu großem Andrang bei der Testatabnahme kommt und keine Nachbesserungen mehr möglich sind.

[2] Webseite für Debian: `http://www.debian.org/`
[3] Webseite der GNU Compiler Collection: `http://gcc.gnu.org/`
[4] Webseite der Microsoft-Entwicklungsplattform: `http://msdn.microsoft.com/vstudio`
[5] Webseite Intel-Entwicklerplattform: `http://software.intel.com/intel-composer-xe`
[6] Webseite des IT Centers der RWTH Aachen: `http://www.itc.rwth-aachen.de/`
[7] Webseite für MinGW: `http://www.mingw.org/`
[8] Webseite für clang: `http://clang.llvm.org/`

4 Arbeiten mit LINUX

Unter Linux stehen Ihnen eine Vielzahl von graphischen Benutzeroberflächen zur Verfügung. Standardmäßig kommt im CIP-Pool KDE zum Einsatz, Sie können aber auch Gnome oder die leichtgewichtigere Desktopumgebung Xfce benutzen.

Viele Aufgaben lassen sich aber komfortabler über ein Terminal in einer textbasierten Benutzeroberfläche, der sogenannten Shell, erledigen. Sie finden sie auf dem Bildschirm unter `konsole` oder `xterm` oder es gibt ein entsprechendes Symbol: den Bildschirm mit Muschel oder Größer-Zeichen. Mit ihm kann eine neue Shell gestartet werden. In dieser können Sie Ihre Befehle eingeben. In der Shell befindet man sich in einem Ordner, dessen Elemente Sie bearbeiten können. Am Anfang befindet man sich immer in dem entsprechenden Home-Verzeichnis. Aus der Shell können auch direkt Programme gestartet werden.

Einige wichtige Befehle sind im Folgenden kurz zusammengefasst:

4.1 Shell-Befehle

`mkdir` *Name*	Erstelle ein Verzeichnis *Name* (make directory).
`rmdir` *Name*	Entferne das (leere) Verzeichnis *Name* (remove directory).
`ls`	Zeige die Dateien im aktuellen Verzeichnis an (list).
`cd` *Name*	Wechsle in das Verzeichnis *Name* (change directory).
`cd ..`	Wechsle in das übergeordnete Verzeichnis.
`cd`	Wechsle ins Home-Verzeichnis.
`cd -`	Wechsle in das zuletzt besuchte Verzeichnis.
`cp` *Quelle Ziel*	Kopiere die Datei *Quelle* nach *Ziel* (copy).
`rm` *Name*	Lösche die Datei *Name* (remove). **Vorsicht: Weg ist weg!**
`mv` *Name Verzeichnis*	Verschiebe die Datei *Name* ins Verzeichnis *Verzeichnis* (move).
`mv` *Name1 Name2*	Benenne die Datei *Name1* in *Name2* um.
`gedit` *Name*	Bearbeite die Textdatei *Name* mit dem Editor `gedit`.
`man` *Befehlsname*	Zeige Hilfe zum Befehl *Befehlsname* an (manual).
`g++ -o` *Prog Name1 Name2 etc.*	Übersetze die C++-Dateien *Name1 Name2 etc.* und füge sie zum Programm *Prog* zusammen.
`ddd | gdb` *Name*	Debugge das Programm *Name*.

4.2 Optionen

Die meisten der Befehle haben eine ganze Reihe zusätzlicher Optionen, mit denen man ihr Verhalten verändern kann. Die Optionen werden fast immer zwischen dem Befehl und dem nachfolgenden Dateinamen eingefügt. Um sie von Dateinamen zu unterscheiden, beginnen sie mit einem Minuszeichen. Beispielsweise zeigt `ls -l` die Dateien mit zusätzlichen Informationen (Größe, Datum etc.) an. Mit `ls -t` kann man sich die Dateien nach dem Datum sortieren lassen. Optionen, die nur aus einem Buchstaben bestehen, kann man oft zusammenfassen: `ls -lt` oder `ls -tl`. Wenn nach einer Option noch eine weitere Angabe (wie zum Beispiel `-o` *Prog* bei `g++`) folgt, dann funktioniert das allerdings nicht.

4.3 Befehle in den Hintergrund schieben

Normalerweise nimmt die Shell erst dann wieder Befehle entgegen, wenn der letzte eingegebene abgearbeitet wurde. Bei Befehlen wie `ls` ist das auch ganz vernünftig. Wenn Sie allerdings einen Editor mit Ihrem selbst geschriebenen Programm gestartet haben, bleibt das Fenster so lange blockiert, bis Sie ihn wieder verlassen haben. Damit das nicht passiert (schließlich wollen Sie ein Programm nach einer Änderung übersetzen und testen, ohne jedes mal den Editor zu verlassen), können Sie stattdessen nach dem Befehl ein `&` eingeben. Durch das Und-Zeichen `&` am Ende wird der Befehl in den „Hintergrund“ verschoben, und die Shell ist sofort für den nächsten Befehl bereit.

5 Editoren zur Textbearbeitung

Um einen Quelltext zu bearbeiten, ist ein Editor hilfreich, der programmiersprachliche Elemente farblich hervorheben kann, evtl. mehrere Dateien nebeneinander aufzeigen kann und einfach zu bedienen ist.

Ein Standardeditor unter Linux ist `gedit`. Aber es gibt auch weitere Editoren, wie z.B: `kwrite`, `kate` oder `scite`, die auf den Praktikumsrechnern vorinstalliert sind.

Die Texteditoren sind meistens recht ähnlich aufgebaut.

Über das Menü am oberen Rand von `gedit` erreicht man alle Befehle. Z.B. Unter *File* gibt es die Befehle, die den Editor bzw. das geöffnete Dokument betreffen (z.B. *Save* zum Abspeichern des gerade bearbeiteten Textes und *Exit* zum Verlassen des Editors). Unter *View* kann man die aktuelle Ansicht anpassen, usw. Die Befehle sind meistens rückgängig zu machen und es lohnt sich, einfach mal auszuprobieren.

Die meisten Befehle besitzen auch Tastenabkürzungen. Das ist recht nützlich, wenn man seinen Schreibfluss an der Tastatur nicht unterbrechen möchte, weil man zur Maus greifen muss.

Die wichtigsten Befehle von `gedit`, zu denen man immer mal wieder greift, sind:

<Ctrl>–n	Neues Editorfenster öffnen.
<Ctrl>–o	Datei öffnen.
<Ctrl>–s	Speichern der Datei.
<Ctrl>–w	Schließen eines Editorfensters.
<Ctrl>–q	Verlassen des Editors.
<Ctrl>–f	Suche in der Datei.
<Ctrl>–g	Weitersuchen, aber nur bei `gedit`; unter Kate und kWrite ist das *F3*
<Ctrl>–h	Suchen und ersetzen, aber nur bei `gedit`; unter Kate und kWrite ist das *<Ctrl>–r*
<Ctrl>–l	Springe zu bestimmter Zeile.
<Ctrl>–c	Markierten Text in die Zwischenablage kopieren.
<Ctrl>–x	Markierten Text ausschneiden und in die Zwischenablage schieben.
<Ctrl>–v	Text aus der Zwischenablage einfügen.
<Ctrl>–z	Änderungen schrittweise rückgängig machen.

Textregionen kann man auch mit der linken Maustaste markieren und dann mittels Rechtsklick ausschneiden, kopieren oder irgendwo anders einfügen.

Es gibt für verschiedene Programmiersprachen verschiedenes Highlighting, welches man unter *View > Highlight Mode* einstellen kann.

Es lohnt sich, die verschiedenen Texteditoren einmal anzuschauen und sich für den zu entscheiden, mit dem man am besten zurecht kommt.

Fortgeschrittene können sich einmal `vi` bzw. `vim`[9] oder `xemacs`[10] ansehen. Diese Editoren sind zwar von der Benutzung her sehr kompliziert, aber wenn man sie einmal beherrscht, geht das Programmieren sehr schnell und effektiv.

6 Versionsverwaltung mit Git

Um die Zusammenarbeit in den Zweiergruppen zu erleichtern, bietet es sich an, den Code unter Versionsverwaltung zu stellen. Besonderer Beliebtheit erfreut sich inzwischen das Versionsverwaltungsprogramm `git`[11].

[9] Webseite zu `vim` (Vi IMproved): `http://www.vim.org/`. Eine Schnellübersicht zum `vim` finden Sie unter `http://tnerual.eriogerg.free.fr/vimqrc-ge.pdf`, eine Kurzanleitung unter `http://lug.fh-swf.de/vim/vim-kurzanleitung.pdf`

[10] Webseite zum XEmacs: `http://www.xemacs.org/`. Die grundlegenden und wichtigsten Tastenkombinationen des `xemacs` sind zum Beispiel unter `http://www.cs.dal.ca/studentservices/refcards/xemacs.pdf` zu finden.

[11] git-Website: `http://git-scm.com/`

Die RWTH bietet unter `https://git.rwth-aachen.de/` einen Gitserver an, auf dem jeder Student bis zu 50 Gitrepositories anlegen kann.

Hinweis: Bitte achtet beim Anlegen eures MaPra-Repositories darauf, dass die Sichtbarkeit auf *Private* eingestellt ist, so dass eure Lösungen nicht von anderen Gruppen angesehen werden können (schließlich soll ja keiner dem Spaß beraubt werden, die Aufgaben selbstständig zu bearbeiten ;-)). Um in einer Zweiergruppe ein gemeinsames Gitrepository zu nutzen, kann einer von beiden ein privates Gitrepository anlegen und dann über die Weboberfläche des Repositories seinen Abgabepartner unter "Members" zur Mitarbeit einladen.

Git bringt eine sehr ausführliche Sammlung von man pages mit, die sich auch gut zum lernen eignet. Neben man pages zu `git`-Unterbefehlen, die mit `git` *`unterbefehl`* `--help` aufgerufen werden können, gibt es auch einige Anleitungen, die man sich mit `git help -g` auflisten lassen kann. Für den Einstieg eignen sich davon `git help tutorial` und `git help everyday`. Außerdem findet man online viele einsteigerfreundliche Anleitungen[12] [13].

Damit ihr nachher sehen könnt, wer welche Änderungen beigetragen hat, solltet ihr vor der ersten Benutzung von `git` euren Namen und eure Emailaddresse konfigurieren:

```
git config --global user.name "Maximiliane␣Musterfrau"
git config --global user.email musterfrau@example.com
```

7 Weitere nützliche Programme

Die folgende Liste enthält einige Programme, die nicht notwendigerweise im mathematischen Praktikum verwendet werden müssen, die aber oft sehr nützlich beim Programmieren bzw. beim Umgang mit Linux sind. Wir geben hier nur eine kurze Beschreibung der einzelnen Programme an. Für eine genauere Beschreibung der Programme und deren Funktionsweise lohnt sich ein Blick in die jeweiligen `man`-pages und auf die in den Fußzeilen angegebenen Websites.

`screen`[14]	kann dazu benutzt werden, eine entfernte Terminal-Sitzung weiterlaufen zu lassen, auch wenn die ssh-Verbindung wegbricht.
`cppcheck`[15]	Statisches Codeanalyseprogramm; findet einige Programmierfehler, die der Compiler nicht entdeckt.

8 Literaturverzeichnis

Literatur

[1] Kruse, C.: *Die sichere Passwort-Wahl.* `http://aktuell.de.selfhtml.org/artikel/gedanken/passwort/index.htm`.

Literatur zur Numerischen Mathematik

[2] Dahmen, W. und A. Reusken: *Numerische Mathematik für Ingenieure und Naturwissenschaftler*. Springer Verlag, Heidelberg, 2. Auflage, 2008.

[3] Deuflhard, P. und A. Hohmann: *Numerische Mathematik I.* de Gruyter, Berlin, 4. Auflage, 2008.

[12] git-Tutorial: `https://www.atlassian.com/git/tutorials/`
[13] kurze Befehlsübersicht: `http://www.cheat-sheets.org/saved-copy/git-cheat-sheet.pdf`
[14] `http://wiki.ubuntuusers.de/screen`
[15] `http://cppcheck.sourceforge.net/`

[4] Schwarz, H.-R. und N. Köckler: *Numerische Mathematik*. B.G. Teubner, Wiesbaden, 5. Auflage, 2004.

[5] Stoer, J.: *Numerische Mathematik I*. Springer Verlag, Heidelberg, 9. Auflage, 2004.

[6] Stoer, J. und R. Bulirsch: *Numerische Mathematik II*. Springer Verlag, Heidelberg, 5. Auflage, 2005.