Aufgabe 1 – *ssh und scp*

Oft genug kommt man in die Situation, dass man sich auf einem anderen Rechner einloggen muss, um dort zu arbeiten. Entweder weil dieser bestimmte Eigenschaften *(bessere/r Prozessor/installierte Programme/...)* erfüllt oder weil auf diesem Dateien oder Programme liegen, auf die man sonst keinen Zugriff hat. Das Kommando, mit dem man sich auf einem anderen Rechner anmelden kann, kennst du bereits aus der Vorlesung.

- Melde dich mit Hilfe von `ssh` auf dem Rechner `faui01` an und schaue, was auf diesem Rechner in `/var/tmp/vorkurs/` liegt.
  Hinweis: Du kommst wieder auf den lokalen Computer zurück, indem du `exit` eintippst.
- Lege auf **deinem** Rechner im Verzeichnis `/var/tmp/` ein neues Verzeichnis mit deinem Loginnamen an und kopiere (mittels `scp`) das, was du auf der `faui01` gefunden hast, in dieses Verzeichnis. *(Tipp: Die Rechnernamen stehen auf den Monitoren oder Rechnern.)*
- Entpacke das Archiv, das du dir gerade kopiert hast, mit dem Befehl `unp` und schau dir die entpackten Dateien an.

Einige der entpackten Dateien kannst du nicht betrachten, da du dazu nicht berechtigt bist. Lesbar machen kannst du sie mit `chmod` (siehe nächste Aufgabe).

Aufgabe 2 – *chmod und Mensaskript*

- Um Dateien für dich lesbar zu machen, musst du für dich (`u`, für user) das read-Flag (`r`) setzen: `chmod u+r <Dateiname>`
- Dies ändert allerdings nur die Rechte für den Besitzer. In dem Fall des entpackten Archivs bist das du. Du kannst nicht die Rechte von Dateien ändern, die dir nicht gehören.
- Manchmal ist es praktisch, alle Dateien in einem Unterordner lesbar zu machen. Wie du dies bewerkstelligst, kannst du in der manpage nachlesen (`man chmod`, Stichwort: *recursive*).

Nun wollen wir `chmod` praktisch anwenden, um ein Skript ausführbar zu machen. Dazu nehmen wir uns das Mensaskript aus der vorherigen Übungsstunde vor.

- Bis jetzt kann man das Skript mit `perl mensa.pl` ausführen, aber dazu muss man natürlich vorher wissen, in welcher Skriptsprache es geschrieben ist. Besser ist es die Datei ausführbar zu machen. Dazu setzt du für dich das *executable*-Flag (`x`): `chmod u+x mensa.pl`.
- Das Ergebnis deiner Berechtigungsänderung kannst du mit `ls -l mensa.pl` betrachten.
- Nun kannst du das Skript direkt mit `./mensa.pl` aufrufen.
- Das Skript kann dir auch den Mensaplan für die einen der nächsten Tage ausgeben, indem du ihm als Argument z.b. die 1 übergibst um den Plan für morgen zu erfahren.
- *Bonusaufgabe* (falls du bereits Erfahrung mit Perl gesammelt hast): Versuche zu verstehen, wie das Skript funktioniert. Was tut der reguläre Ausdruck?

Aufgabe 3 – *Failed Backup*

Im Verzeichnis `/proj/ciptmp/vorkurs/uebung/aufgabe_failedbackup/archive/` findest du ein Archiv mit den Namen `failedbackup.tar.bz2`. In dieses Archiv wurden „versehentlich“ falsche Dateien eingepackt.

Kopiere das Archiv `failedbackup.tar.bz2` in ein temporäres Verzeichnis (z. B. `/var/tmp/<username>`) und entpacke es **dort**.
Wechsle nun in das entstandene Verzeichnis `failedbackup`.

- Wie viele Dateien sind im Verzeichnis `dir2`[1]? (Bitte nicht per Hand zählen[2]!)

Nun wollen wir das Backup bereinigen und ein paar Dateien löschen:

- Lösche alle Dateien, die mit `old_` beginnen, aus dem Verzeichnis `dir2`.
- Lösche alle Dateien, die mit `arts` beginnen, aus dem Verzeichnis `dir1`[3].
- Lösche alle Dateien, die auf `.bak` enden, aus den *beiden* Verzeichnissen `dir1` und `dir2`.
- Wie viele Dateien sind nun noch in den beiden Verzeichnissen[4]?

Um herauszufinden, ob du die richtigen Dateien entfernt hast, kannst du dir eine Liste mit den Dateinamen erzeugen und eine Prüfsumme darüber berechnen lassen.

- Wechsle ins Verzeichnis `failedbackup`.
- Verwende `find`, um alle Dateien anzeigen zu lassen. Sortiere die Ausgabe mit `sort` (um gleiche Sortierung zu erreichen, verwende die Argumente `-fd`) und leite die Ausgabe an das Programm `md5sum`[5] weiter.

Die Aufgabe ist korrekt erledigt, wenn die Prüfsumme `274455e39230db7fb0d2514ea2302485` ist. Solltest du eine andere Prüfsumme haben, dann prüfe, ob noch zu löschende Dateien existieren. Oder hast du vielleicht versehentlich eine neue Datei angelegt?

Erstelle nun aus dem Verzeichnis ein neues Archiv mit dem Namen `fixedbackup_ok.tar.bz2`.

Aufgabe 4 – *Die Kontrolle behalten*

Starte in einer Konsole die Vorlesungsfolien mit `evince`. Wechsle zurück zur Konsole und stoppe das Programm mit `Ctrl-Z`. Schicke `evince` nun in den Hintergrund (`bg`). Nun kannst du deine Konsole für ein anderes Programm benutzen. Starte zum Beispiel `icedove`. Schicke dieses nun auch in den Hintergrund. Mit `jobs` kannst du jederzeit sehen, welche Jobs du gerade laufen hast.

[1] Ich biete 2298.
[2] Tipp: `ls` und `wc` geschickt „verbinden“
[3] Das klappt nicht? Da fehlen wohl irgendwelche Rechte. Nur welche?
[4] Ich biete 1529 und 766.
[5] `md5sum` liest von stdin Daten ein und bildet eine Prüfsumme über die Daten.

Aufgabe 5 – *Kill me!*

Führe zuerst das Programm `/proj/ciptmp/vorkurs/killme` aus. Öffne danach eine neue Shell und versuche das Programm zu beenden. *Tipp: Lies dir die Ausgabe des Programms durch!*

Als nächstes führst du das Programm `/proj/ciptmp/vorkurs/tetris` aus. Lässt sich das Programm wie das `killme`-Programm beenden? Wenn nein, dann erinner dich an die Vorlesungfolien[6].

Und die Moral von der Geschicht': Nicht einfach Programme von anderen Leuten ausführen, wenn man nicht weiß, was sie tun. Manchmal trügt der Schein. ☺[7]

Aufgabe 6 – *Quota*

In deinem *Home*-Verzeichnis hast du anfangs 200 MB (als Informatik-Student 400 MB) Speicherplatz zur Verfügung – und der ist überraschend schnell voll. Vor allem wenn du das *tetris*-Skript aus Aufgabe 5 ausgeführt hast. Falls du die vorherige Aufgabe übersprungen hast, so bearbeite nun wenigstens deren zweiten Teil (*tetris*).

Um zu überprüfen, wie viel Platz noch frei ist, kannst du (natürlich in einer Shell) den `quota`-Befehl verwenden:

- blocks: der belegte Speicher (in KiB)
- quota: der zur Verfügung stehende Speicher (in KiB)

Schau in der Manpage von `quota` nach, wie man die Ausgabe auf menschenlesbar („human-readable“) stellen kann!

Falls du herausgefunden hast, dass du zu viel Speicher belegst ☺, kannst du mit dem Programm `du` (für *disk usage*) nachschauen, welche Verzeichnisse wie viel Speicherplatz belegen. Schau nach, welches Verzeichnis in deinem Home den meisten Speicherplatz verbraucht. Verwende die Option `--max-depth=1`, um nur die Verzeichnisse direkt in deinem Home anzuzeigen.

Auch wenn man mit der `--max-depth`-Option die Datenflut schon deutlich reduzieren kann, ist die Ausgabe gerade bei großen Verzeichnissen immer noch recht unübersichtlich. Um schnell große Dateien zu finden, wäre es also hilfreich, die vielen Informationen von `du` sortieren zu können. Filtere die Ausgabe von `du` durch `sort -n` (`-n` sortiert die Daten nicht einfach alphabetisch, sondern erkennt auch Zahlen[8]). Wenn du willst, kannst du die sortierte Ausgabe noch durch ein weiteres Programm *pipen*, wie z. B. `tail` oder `less`.
Beachte: `sort` kann die menschenlesbare Ausgabe von `du` nicht korrekt sortieren!
Alternativ kannst du auch das Programm `ncdu` verwenden (mit `q` beendet man es wieder), das dir eine sortierte Anzeige der Speicherplatzfresser im momentanen Verzeichnis anzeigt und es dir auch erlaubt, durch Unterverzeichnisse zu navigieren.

Nun möchtest du die Dateien, die das amoklaufende *tetris*-Programm erstellt hat, wieder löschen. Die Dateien sind einigermaßen erkennbar benannt, also sollte es für dich nicht all zu schwer sein, sie zu identifizieren.

Ein Stück intuitiver ist das Tool `baobab`, welches die Größe von Verzeichnissen grafisch darstellt. Am besten ruft man `baobab` aus einer Shell auf und übergibt ihm gleich das Verzeichnis, das man

[6] Tipp: `-9`

[7] Falls es dich interessiert, beide Programme sind Skripte, die du mit `cat` betrachten kannst.

[8] `sort` muss natürlich warten, bis alle Daten vorhanden sind, es kann also ein bisschen dauern.

untersuchen will, als Parameter (z. B. ~ für das Home-Verzeichnis).

Da man sich nicht mehr an den PCs anmelden kann, wenn man sein Quota überschritten hat, kann man `baobab` im Ernstfall nicht verwenden. In diesem Fall kann man mit `Strg+Alt+F1` auf ein anderes *virtuelles Terminal* wechseln und sich dort anmelden und ein paar Dateien löschen. Ist genug Platz, **melde dich wieder ab** (`exit`) und wechsle mit `Strg+Alt+F7` wieder zurück zur grafischen Oberfläche.

Aufgabe 7 – *Feedback*

Wir würden uns über Feedback zu unserem Einführungskurs freuen. Dazu gibt es unter `http://fsi.informatik.uni-erlangen.de/feedback` ein kleines Evaluationssystem, in dem jeder anonym seinen Kommentar zur Veranstaltung abgeben kann. Nimm dir bitte die Zeit und gib uns Feedback.

Anregungen für Feedback:
Wie hat dir die Vorlesung gefallen? Wie waren die Übungen? Welchen Eindruck haben die Übungsleiter auf dich gemacht? Könntest du den Erstis des nächsten Jahres den Kurs empfehlen?

Zusatzaufgaben

Linux für Zuhause

Nachdem du zu der Überzeugung gekommen bist, dass Linux toll ist ☺, möchtest du vielleicht auch auf deinem heimischen PC ein bisschen mehr mit einer Linux-Distribution herumspielen. Mittlerweile gibt es etliche Distributionen, die so ausgereift und benutzerfreundlich sind, dass der Umstieg von (bzw. Parallelbetrieb mit) Windows kein detailliertes Fachwissen mehr erfordert, sondern eine richtig simple Angelegenheit ist.

Ein besonders populärer Vertreter ist Ubuntu Linux – eine stark aufgebohrte Variante von Debian (der Distribution, die im CIP installiert ist). Ubuntu gilt als besonders einsteigerfreundlich und wird deshalb von uns empfohlen, wenn du noch keine erweiterten Linux-Vorkenntnisse mitbringst. xubuntu, eine Ubuntu-Variante, liefert die Desktopumgebung XFCE4 aus, die du auch im CIP vorfindest und ist resourcenschonender als normales ubuntu.

xubuntu kann kostenlos aus dem Internet heruntergeladen und auf CD gebrannt oder wie hier beschrieben auf einen leeren USB-Stick kopiert werden. Dabei handelt es sich um eine sogenanntes Live-System, d. h. du kannst das System komplett vom Stick starten, ohne dass zunächst eine Installation nötig ist. Wenn dir xubuntu gefällt, kannst du es endgültig auf deiner Festplatte installieren - eine bereits vorhandene Windows-Installation würde dabei nicht verloren gehen!

Für die folgenden Schritte benötigst du einen komplett leeren USB-Stick mit einer Mindestgröße von 1GB.
!! Die Anleitung wird alle Daten die noch auf dem Stick sind vernichten !!

Das Installationsabbild findest du fuer Ubuntu als auch Xubuntu in
`/proj/ciptmp/vorkurs/uebung/ubuntu`

Mit dem Befehl dd werden Daten vom infile ins outfile kopiert, in diesem Fall ist das outfile ein USB-Stick.
ACHTUNG: egal was bei of= angegeben ist wird komplett ueberschrieben.

```
cd /proj/ciptmp/vorkurs/uebung/ubuntu
dd if=xubuntu-14.04.1-desktop-amd64.iso of=/dev/sdb bs=1M
```

Das Outfile muss das Device-File des USB-Sticks sein, i.A. wird das im CIP meist /dev/sdb sein.

Vom so erstellten Live-USB-Stick kann nun gebootet werden, das muss evt. im BIOS oder Bootmenu eingestellt werden.

Falls du dir unsicher bist und dir Hilfe bei der Installation wünschst dann komm doch zur Linux-Install-Party (http://fsi.cs.fau.de/linuxinstall), welche dieses Jahr am Mittwoch, den 5.11.2014 ab 14:00 rechts neben dem CIP2.

Hinweis: Da das ciptmp-Verzeichnis wie in der Vorlesung erwähnt per Automounter eingebunden ist, wird es unter Umständen zunächst nicht in `/proj` angezeigt. Sollte das der Fall sein, gib einfach den kompletten Pfad `/proj/ciptmp/vorkurs/uebung/ubuntu` in das Textfeld ein.

Aliase

Manchmal erleichtert es die Arbeit massiv, wenn man lange Befehle, die man häufig tippt, abkürzen kann. Dazu kann man in der Bash *Aliase* anlegen. Aliase sind einfach nur Abkürzungen. Um alle schon vordefinierten Aliase zu betrachten, reicht es, wenn du `alias` eintippst. Mit `alias` kann man auch neue Abkürzungen definieren. Dazu verwendet man die Form `alias <Abkürzung>='<Befehl>'`. Ein Beispiel wäre `alias l='ls -l'`, was ein neues Alias **l** definiert.

So definierte Aliase gelten allerdings nur für die aktuelle Shell und sind nach deren Beenden nicht mehr verfügbar. Willst du dauerhaft ein Alias anlegen, schreibst den `alias`-Befehl ans Ende deiner *~/.bashrc*. Diese Datei wird von jeder interaktiven bash-Shell beim Starten ausgeführt.

Mail auf der Konsole

Um schnell mal eine Mail von der Konsole aus zu schreiben, gibt es das suggestiv benannte Programm `mail`. Um eine Mail zu schreiben, einfach `mail <email-adresse>` tippen und schon geht's los. Wenn du fertig bist mit Tippen, einfach eine Zeile eingeben, in der **nur ein Punkt** ist.

Du kannst auch jedem anderen Benutzer im cip eine Mail schreiben, indem du seinen Benutzernamen als Mail-Adresse benutzt (ohne @). Deinen eigenen Benutzernamen bekommst du durch den Befehl `whoami`. Um direkte Mails im CIP zu lesen, einfach `mail` ohne Argumente starten und die Mail durch Eingabe der Nummer auswählen. Schreibe nun mit jemandem in deiner Umgebung ein paar Mails hin und her.

Willst du es noch etwas komfortabler Mails lesen, aber trotzdem das ganze noch per ssh machen können, dann kannst du dir mal das Programm `mutt` anschauen.

Noch mehr Aufgaben

Du hast immer noch Interesse an weiteren Übungsaufgaben? Dann guck doch mal unter folgender Adresse vorbei:
`https://fsi.informatik.uni-erlangen.de/dw/informationen/vorkurs/aufgabensammlung/`