GRA Intro: Setup

Da viele von euch jetzt in GRA das erste Mal mit einer UNIX/Linux-Umgebung arbeiten werden, hier eine kleine Einführung.
Ich kann und werde keinen 1:1 Tech-Support für jeden von euch leisten können, schließlich bin ich selbst Vollzeistudent :) Bei Fragen könnt ihr aber jederzeit in den öffentlichen Zulip-Streams zu GRA schreiben.

1 Betriebssystem

1.1 Windows

Bitte installiere Windows Subsystem for Linux (WSL). Du kannst dann jederzeit WSL einfach vom Startmenü aus starten und erhälst ein voll funktionsfähiges Ubuntu. Du kannst auf die in deiner WSL-Umgebung gespeicherten Daten über den Datei-Explorer zugreifen, dafür gibts den Befehl „`explorer.exe .`“ (den letzten Punkt nicht vergessen).
Ich würde stark davon abraten, alles nativ in der Windows Command Line versuchen zum Laufen zu bringen.

1.2 Linux

Falls bereits irgendeine Linux-Distro auf deinem Rechner läuft, hast du alles richtig gemacht.

1.3 MacOS

Ich habe selbst nie mit MacOS gearbeitet, aber afaik sollte dieses Betriebssystem auf UNIX basieren, d.h. das meiste sollte ohne Probleme laufen. Wichtig für spätere teile des Kurses einige der Hardwarenahen Befehle e.g. Intel SIMD (SSE 1 - 4) Instuktionen sind auf den neuen M-Prozessoren nicht verfügbar bearbeiten und debuggen dieser Aufgaben am besten auf der Rechnerhalle über SSH.

2 IDE/Editor

- Lightweight und gute Integration, aber umständliches Debugging: VSCode
- Sehr starke Debugging-Umgebung (ähnlich zu IntelliJ), aber manchmal ein bisschen kniffliges Setup mit Makefiles/WSL: CLion (Lizenz dafür gibt's als TUM-Studi kostenlos)
- Für ~~Masochisten~~ Profis: VIM/Emacs/Helix...

3 Einige nützliche Befehle

`cd <path>`	Wechselt das current working directory zu `<path>`
`ls -la`	Listet alle Dateien im CWD auf. (`-l` für erweiterte Infos, `-a` um auch versteckte Dateien anzuzeigen)
`rm <file>`	Entfernt die Datei `file`
`code .`	Startet VSCode im CWD
`make`	Startet die in der Makefile vorgebene Routine, falls sich die Ausgangsdateien geändert haben
`make clean`	Das selbe wie `make`, nur unabhängig davon, ob sich die Ausgangsdateien geändert haben oder nicht
`chmod u+x <file>`	Markiert die Datei `file` als ausführbar

`./<file>`	Führt eine Datei aus
`nano <file>`	Simpler Texteditor
`ssh <rbg-kennung>@ lxhalle.in.tum.de`	SSH-Login zur Rechnerhalle
`git status`	Zeigt die aktuell getrackten Dateien an
`git commit --allow-empty -m ""`	Erstellt einen leeren Commit, sodass nach einem Push auf Artemis neu gebuildet wird
`git diff`	Zeigt Dateiänderungen im aktuellen Git-Repo an
`ctrl + c`	Kein Befehl, sondern eine Tastenkombination: Unterbricht die Ausführung des aktuellen Prozesses
`sl`	Überlebenswichtiger Befehl