Programmierkurs
Installationsanleitungen

Dr. Ole Klein

Interdisziplinäres Zentrum für Wissenschaftliches Rechnen
Universität Heidelberg

6. November 2020

Ihre Möglichkeiten

WSL-Installation (Windows 10)
- WSL installieren
- Ubuntu einrichten
- Arbeitsumgebung einrichten
- SDL-Bibliothek für fortgeschrittene Übungen

Einrichtung einer virtuellen Maschine
- Installationsanleitung
- Problembehandlung
- Nutzung

Ihre Möglichkeiten

Grundsätzlich:

- die Vorlesung orientiert sich an Unix (Linux,macOS)
- Kein Support für das Programmieren unter Windows, da wir für die Vielzahl an Konfigurationen / Softwarelösungen nicht die nötigen Ressourcen haben!

Sie können wählen:

- Installation des WSL (Windows Subsystem for Linux)
- Verwendung einer virtuellen Maschine
- Installation bzw. Verwendung eines Linux- oder macOS-Systems
- Verwendung von Windows "auf eigene Faust"

Ihre Möglichkeiten

Nutzern von Windows 10 empfehlen wir das WSL:

- Komfortabel und gut in Windows integriert
- Verhältnismäßig einfach zu installieren

Wer das WSL nicht nutzen kann oder will, kann eine virtuelle Maschine verwenden:

- Etwas weniger komfortabel (z.B. Zugriff auf Dateien von Windows aus)
- Einfache Installation und Erstellung von Sicherungskopien

Wer sich ein Linux parallel installieren möchte, oder absichtlich unter Windows programmieren möchte, kann das trotzdem gerne auf eigene Verantwortung machen!

WSL installieren

- Folgen Sie den Anweisungen auf docs.microsoft.com/en-us/windows/wsl/install-win10
- Für uns reichen die Schritte 1 und 6, der Rest ist optional.
- Starten Sie eine PowerShell als Administrator (z.B. per Rechtsklick im Startmenü)
- Schritt 1: führen Sie den folgenden Befehl aus (Achtung, das ist eine einzelne Zeile):

```
dism.exe /online /enable-feature
    /featurename:Microsoft-Windows-Subsystem-Linux
    /all /norestart
```

- Schritt 6: Installieren Sie eine Linux-Distribution aus dem Microsoft Store, am besten Ubuntu
- Führen Sie einen Neustart des Computers durch

Ubuntu einrichten

- Starten Sie Ubuntu, und richten Sie einen Account mit Passwort ein (kann der Einfachheit halber mit Ihren Daten für Windows übereinstimmen)
- Aktualisieren Sie die Paketverwaltung, und installieren Sie die nötigen Updates:

```
sudo apt update
sudo apt upgrade
```

- `sudo` führt unter Linux Befehle als Administrator aus. Ab und zu müssen Sie dafür das oben vergebene Passwort eingeben.
- Installieren Sie die Compiler `g++` und `clang++`, den Debugger `gdb`, und das Buildsystem `cmake`:

```
sudo apt install build-essential clang cmake
```

Ubuntu einrichten

- Damit ist die Installation im Prinzip bereits abgeschlossen, wir konfigurieren aber noch kurz ein paar Kleinigkeiten.
- Ihr Homeverzeichnis in Ubuntu entspricht unter Windows einem Ordner irgendwo innerhalb des Ubuntu-Installationsverzeichnisses:
  - Schwer zu finden und zu nutzen
  - Problematisch bei Updates bzw. Re-/Deinstallation
- Legen Sie symbolische Links auf die bekannten Windows-Ordner an:

```
ln -s /mnt/c/Users/<username>/Documents/ Documents
ln -s /mnt/c/Users/<username>/Desktop/ Desktop
```

Ersetzen Sie dabei `<username>` durch Ihren Windows-User, und `"Documents"` evtl. durch `"Dokumente"` etc. (kommt leider auf Ihre Spracheinstellungen an).

Ubuntu einrichten

- Dateien in diesen Ordnern (und Unterordnern) sind jetzt in beiden Betriebssystemen zugreifbar
- Legen Sie daher Dateien und Verzeichnisse nur in diesen beiden Ordnern an, nicht direkt im Homeverzeichnis!
- Sie können jetzt Dateien komfortabel unter Windows erstellen, modifizieren und löschen, und unter Ubuntu Programme kompilieren und ausführen

Warnung

Ein unvorsichtiges `rm` in Ubuntu kann jetzt Ihre Dokumente in Windows löschen! Entfernen Sie alte Dateien daher lieber graphisch unter Windows!

Arbeitsumgebung einrichten

Sie können zwar prinzipiell Ihre Programme von Windows aus erstellen, wir installieren aber im folgenden eine Arbeitsumgebung unter Ubuntu.

- Sie können dann alle Schritte komplett unter Ubuntu durchführen
- Für die fortgeschrittenen Übungen müssten Sie das meiste davon sowieso umsetzen

Installieren Sie die Entwicklungsumgebung `geany` und den Terminalemulator `sakura`:

```
sudo apt install geany sakura
```

Hinweis

Neben den hier vorgeschlagenen Paketen können Sie sich natürlich auch über Alternativen informieren und diese installieren.

Arbeitsumgebung einrichten

- Installieren Sie `VcXsrv` von der Seite sourceforge.net/projects/vcxsrv/
- Dabei handelt es sich um einen X-Server (das Programm, das unter Linux die Darstellung von graphischen Elementen wie Fenstern verwaltet)
- Starten Sie das Programm, die Standardeinstellungen sind dabei ausreichend. Wenn Sie die Konfiguration speichern, wird der X-Server in Zukunft automatisch mit Windows gestartet.
- Führen Sie danach die folgenden Befehle aus, um Ihren X-Server in Ubuntu bekannt zu machen:

```
# make VcXsrv known to Ubuntu
echo "export DISPLAY=localhost:0.0" >> ~/.bashrc
# fix rendering of OpenGL
echo "export LIBGL_ALWAYS_INDIRECT=1" >> ~/.bashrc
# execute changes in current session
. ~/.bashrc
```

Arbeitsumgebung einrichten

Starten Sie geany:

```
geany &
```

Und fügen Sie in das Fenster den folgenden Code ein:

```
#include <iostream>

int main(int argc, char** argv)
{
  std::cout << "hello world!" << std::endl;
  return 0;
};
```

Machen Sie sich mit den Steuerelementen von geany vertraut (Speichern, Kompilieren, Ausführen). Wenn alles klappt, sollte sich ein schwarzes Fenster mit Text öffnen, dass die Ausgabe des Programms zeigt.

Installation der SDL-Bibliothek

Im Verlauf der Übungen werden wir als Beispiel einer verhältnismäßig einfachen externen Bibliothek das `Simple DirectMedia Layer` (SDL) verwenden.

- Installieren Sie die SDL-Bibliothek:

```
# install SDL2 library for advanced exercises
sudo apt install libsdl2-dev
```

- Teilen Sie `geany` in den Kompilieroptionen mit, dass gegen die SDL-Bibliothek gelinkt werden soll:

```
g++ -Wall -o "%e" "%f" -lSDL2
```

(Beachten Sie, dass solche Linkeroptionen immer ganz rechts stehen müssen.)

Installation der SDL-Bibliothek

Fügen Sie die folgenden Codezeilen in geany ein (alle Folien nacheinander in eine Datei):

```
#include <SDL2/SDL.h>
#include <iostream>

int main(int argc, char** argv)
{
  // try to initialize SDL2
  if (SDL_Init(SDL_INIT_VIDEO) < 0)
  {
    std::cerr << "could not initialize SDL2: "
      << SDL_GetError() << std::endl;
    return 1;
  }

  // create window of size 640 x 480
  const int width = 640;
  const int height = 480;
  SDL_Window* window = SDL_CreateWindow("helloWorld",
    SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
    width, height, SDL_WINDOW_SHOWN);
```

```
  if (window == nullptr)
  {
    std::cerr << "could not create window: "
      << SDL_GetError() << std::endl;
    return 1;
  }

  // access canvas in window
  SDL_Surface* surface = SDL_GetWindowSurface(window);
  if (surface == nullptr)
  {
    std::cerr << "could not get surface: "
      << SDL_GetError() << std::endl;
    return 1;
  }

  // fill with white color
  SDL_FillRect(surface, nullptr,
    SDL_MapRGB(surface->format, 0xFF, 0xFF, 0xFF));

  // draw a small green rectangle in center
  SDL_Rect rect;
  rect.x = width/2 - 16;
  rect.y = height/2 - 16;
  rect.w = rect.h = 32;
  SDL_FillRect(surface, &rect,
    SDL_MapRGB(surface->format, 0x00, 0xFF, 0x00));
```

```
    // show the resulting surface and wait 5s
    SDL_UpdateWindowSurface(window);
    SDL_Delay(5000);

    // destroy window and quit SDL2
    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
}
```

Wenn alles korrekt funktioniert, sollte das Kompilieren und Ausführen dieses Programms ein kleines Fenster öffnen, in dem Sie auf weißem Grund ein grünes Quadrat präsentiert bekommen. Es schließt sich dann nach fünf Sekunden automatisch.

Virtuelle Maschine einrichten

Alternative zum WSL: Virtuelle Maschine unter VirtualBox

Voraussetzungen:

- 20 GB Festplattenspeicher (vermutlich genügen $\sim$ 15 GB)
- 4 GB RAM, besser sind 6 GB oder mehr
- Installieren Sie VirtualBox 6.1 von der Seite www.virtualbox.org
- Nach dem Starten von VirtualBox laden Sie bitte das VirtualBox Extension Pack herunter und installieren es: www.virtualbox.org/wiki/Downloads

VM — Anleitung

- Laden Sie die virtuelle Maschine von unserem Server:
  conan.iwr.uni-heidelberg.de/files/teaching/ipk-2020/ubuntu-ipk-2020.zip
  Entpacken Sie die vbox-Datei `ubuntu-ipk-2020.vbox`, danach können Sie die zip-Datei entfernen.

  Vorsicht

  Diese Dateien sind jeweils mehrere GB groß!

- Starten Sie VirtualBox. Unter "Maschine — Hinzufügen" können Sie zur entpackten Datei gehen und sie hinzufügen.
- Im Hauptmenü können Sie die Maschine nun auswählen und starten.

Zugangsdaten

Nutzer: `ipk`, Passwort: ebenfalls `ipk`

VM — Probleme

- Falls Sie die virtuelle Maschine nicht starten können, könnte es sein, dass Sie Virtualisierung im BIOS aktivieren müssen. Gehen Sie in Ihr BIOS und halten Sie Ausschau nach `Virtualization Support` oder `VT-x`. Die genaue Bezeichnung hängt leider vom Hersteller Ihres Motherboards ab.
- Unter den Einstellungen der virtuellen Maschine können Sie auswählen, wie viel Arbeitsspeicher der virtuellen Maschine zur Verfügung steht. Anfangs ist es auf 2 GB RAM eingestellt. Passen Sie diesen Wert falls nötig an Ihren verfügbaren Arbeitsspeicher an.

VM — Benutzung

- In der virtuellen Maschine ist ein Ubuntu installiert, mit allen für die Vorlesung benötigten Packeten.
- Zum Programmieren können Sie einen der einfachen Texteditoren (`gedit`, `kate`), eine der IDEs (`geany`, `qtcreator`, `gnome-builder`) oder etwas anderes verwenden.
- Der Nutzer `ipk` hat `sudo` Rechte, Sie können die virtuelle Maschine also ganz nach Ihren Bedürfnissen anpassen.
- Wer technisch versiert ist, kann durchaus spezielle Ordner zum Dateiaustausch mit Windows anlegen. Ansonsten empfehlen wir, ggf. einen Onlinedienst zum Austausch zu verwenden (heiBOX, Dropbox, Google Drive, etc.).