1 Übungsaufgabe #1: Arbeitsumgebung einrichten

In dieser Aufgabe werdet ihr eure Arbeitsumgebung für die weiteren Aufgaben in PASST einzurichten. Dabei lernt ihr den Umgang mit den notwendigen Werkzeugen.
Die Erkenntnisse und Ergebnisse dieser Aufgabe dienen als Basis für die folgenden Praktikumsaufgaben.

1.1 Erstellen eines lauffähigen Linux-Systems

Damit das Arbeiten mit dem Linux-Kern möglichst einfach und effizient vonstatten geht, soll dieser in einer virtuellen Maschine (`VM`) getestet werden. Wir legen euch die Verwendung von `qemu/kvm` nahe.
Da zu einem lauffähigen System nicht nur ein Linux-Kern gehört, soll im ersten Schritt in der `VM` eine Laufzeitumgebung (Dateisystem, Gerätedateien, Systemprogramme wie `init`, `sh`, `ls`, ...) bereitgestellt werden, die dann durch euren eigenen, selbst übersetzten Kernel benutzt werden kann.
Als Basis soll euch hierbei eine Installation der Linuxdistribution Debian dienen. Am Ende dieser Teilaufgabe sollt ihr ein in einer `VM` laufendes Debian-Linux haben. Im nächsten Schritt ersetzt ihr dann den Distributions-Kern durch einen selbst übersetzten.

1.2 Linux-Kernelquellen beschaffen, konfigurieren und übersetzen

Benutzt das Versionskontrollsystem `git`, um die im Verzeichnis `/proj/i4passt/kernel/linux-stable` bereitgestellten Linux-Quellen zu klonen. Als nächsten Schritt müsst ihr eine passende Kernel-Konfiguration erstellen. Diese muss zum einen die passenden Treiber für die "Hardware" enthalten. Zum anderen sollen Debugging-Features (`kgdb`) aktiviert werden, damit man den Kernel mit Hilfe des `gdb` debuggen kann.
Den konfigurierten Kern müsst ihr nun noch übersetzten. Da ihr im Laufe des Praktikums den Kernel noch öfters übersetzen werdet, soll der Übersetzungsvorgang möglichst schnell vonstatten gehen. (→ `ccache`)

1.3 Booten der `VM` mit eigenem Kernel

Ist der Kern übersetzt, soll dieser in der in 1.1 erstellten Umgebung gestartet werden. In der Tafelübung wurde dazu das `boot.sh`-Skript vorgestellt, welches die notwendigen Optionen für `kvm/qemu` kapselt. Allerdings sollt ihr das nicht einfach nur benutzen, sondern die Wirkungsweise und den Sinn der jeweiligen Optionen in der Dokumentation zu `kvm/qemu` nachschlagen, so dass ihr sie uns auch erklären könnt.

1.4 Debugging mit `kgdb`

Schlussendlich sollt ihr nun mit Hilfe des `kgdb` einen Breakpoint innerhalb eures Kernels setzen und ihn damit "im Betrieb" anhalten.

Aufgaben:

→ Einrichten einer Umgebung in einer `VM`.

→ Klonen, konfigurieren und übersetzen des Kernels.

→ Booten und anhängen an einen laufenden Linux-Kern mit `gdb`.

Hinweise:

- Mit Hilfe der `make`-Option `'O=<directory>'` könnt ihr ein extra Ausgabeverzeichnis für den Build-Prozess angeben. Insbesondere kann man somit mehrere Konfigurationen 'nebeneinander' bauen, wenn man das `'O=<directory>'`-Flag auch bei `make menuconfig` mit angibt.
- Bitte baut einen 64-Bit Kernel (Architektur: `amd64`).
- Abschluss der Aufgabe durch Vorführung des lauffähigen Systems mit eigenem Kernel und einiger Debugging-Schritte in einer Recherübung.

1.5 Abgabe: am 13. Mai 2013