Distcc

Allgemeines und Installation

Distcc ist ein Programm, welches sich um die Verteilung von Kompilierungsprozessen an teilnehmende Rechner kümmert. Es besteht aus einem Serverteil, distccd und einem Clientprogramm, distcc. Mit etwas Konfigurationsaufwand funktioniert distcc mit ccache, Portage und Automake zusammen.

Um distcc einsetzen zu können, müssen alle Computer im Netzwerk die gleiche GCC-Version benutzen. Sie können verschiedene 3.3.x Versionen (wobei x variiert) verwenden, aber ein gleichzeitiger Gebrauch von 3.3.x und 3.2.x kann zu Fehlern bei der Kompilierung oder Ausführung von Programmen führen.

Distcc kommt mit einem grafischen Überwachungstool, um einzelne Aufgaben, die an andere PCs verteilt werden, aufzuzeichnen. Als Gnome Benutzer sollten sie 'gnome' in Ihren USE Flags setzen. Verwenden Sie kein Gnome, wollen jedoch die Funktionalität des grafischen Tools nicht missen, dann sollten Sie 'gtk' in den USE Flags gesetzt haben.

```
emerge -va distcc
```

Portage so einrichten, dass es Distcc benutzt

Um heraus zu finden was für cflags gesetzt werden müssen gibt man folgenden Befehl ein

```
gcc -Q --help=target -march=native | grep march
```

```
nano /etc/make.conf
```

Dort fügt man distcc zu den Features hinzu.

```
FEATURES="distcc"
```

Dies wäre also der Clientrechner der die Kompileranfrage ins Netz sendet. Die Liste der Hosts, bearbeitet man mit **distcc-config**.

```
distcc-config --set-hosts "hostname1 hostname2 hostname3"
```

Man kann natürlich auch gesamte Netze eingeben.

Passen Sie /etc/conf.d/distccd an Ihre Bedürfnisse an. Vergessen Sie nicht, durch die –allow Anweisung die Hosts anzugeben, denen Sie vertrauen. Noch mehr Sicherheit erhalten Sie durch Einsatz der –listen Anweisung, die dem distcc Dämon mitteilt, auf welche IP-Adressen er lauschen soll. Starten Sie jetzt den distcc Dämon auf allen teilnehmenden Computern:

```
rc-update add distccd default
```

```
etc/init.d/distccd start
```

Monitoring gibt es mit

```
distccmon-text
```

DistCC Cross-Compiling

Distcc ist ein Werkzeug, mit dem Sie die Last der Software-Kompilierung über mehrere vernetzte Rechner verteilen können. Solange die vernetzten Rechner alle die gleiche Toolchain für die gleiche Prozessorarchitektur benutzen, benötigen Sie keine spezielle Konfiguration des distcc. Aber was machen Sie, wenn Sie für eine abweichende Architektur auf verschiedenen Computern kompilieren möchten? Diese Anleitung wird Ihnen zeigen, wie Sie distcc konfigurieren müssen, um für unterschiedliche Architekturen zu kompilieren.

Emergen der notwendigen Werkzeuge

Als erstes müssen Sie crossdev und distcc auf allen Rechnern, die im Kompilierungsprozess eingebunden werden sollen, emergen. Crossdev ist ein Werkzeug, das das Erstellen von architekturübergreifenden Toolchains vereinfacht. Es wurde ursprünglich von Joshua Kinard geschrieben und dann von Mike Frysinger von Grund auf neu entwickelt. Seine Benutzung ist unkompliziert: crossdev -t sparc erstellt eine vollständig auf die Sparc-Architektur abzielende Cross-Toolchain. Diese beinhaltet binutils, gcc, glibc und linux-headers. Wenn Sie weitere Hilfe benötigen, versuchen Sie crossdev –help auszuführen. Selbstverständlich benötigen Sie die passende Cross-Toolchain auf allen Hilfsrechnern.

Architektur-spezifische Anmerkungen

Wenn Sie zwischen verschiedenen Subarchitekturen für Intel x86 (z.B. i586 und i686) cross-kompilieren, müssen Sie trotzdem eine vollständige Cross-Toolchain für den gewünschten CHOST erstellen, ansonsten wird die Kompilierung fehlschlagen. Das liegt daran, dass i586 und i686 genau genommen verschiedene CHOSTs sind, abgesehen von der Tatsache, dass beide als „x86“ wahrgenommen werden. Behalten Sie das bitte im Hinterkopf, wenn Sie Ihre Cross-Toolchains erstellen. Zum Beispiel, wenn der Zielrechner ein i586 ist, heißt das, dass Sie i586 Cross-Toolchains auf Ihren i686 Hilfsrechnern erstellen müssen.

Konfiguration von distcc um fehlerfrei Cross-Compiling zu nutzen

In der Standardeinstellung von distcc wird Cross-Compiling nicht ordnungsgemäß funktionieren. Das Problem ist, dass viele Builds einfach gcc aufrufen, anstatt des vollständigen Compilernamens (z.B. sparc-unknown-linux-gnu-gcc). Wenn dieser Kompilierungsvorgang dann auf die distcc Hilfsrechner verteilt wird, wird der systemeigene Compiler aufgerufen, anstelle Ihres brandneuen Cross-Compilers. Glücklicherweise gibt es eine Lösung für dieses kleine Problem. Sie benötigen nur ein Wrapper-Skript und ein paar Symlinks auf dem Rechner, der emerge ausführen wird. Ich werde meinen Sparc-Rechner als Beispiel nehmen. Wo auch immer Sie jetzt sparc-unknown-linux-gnu sehen, müssen Sie

Ihren eigenen CHOST (zum Beispiel x86_64-pc-linux-gnu für einen AMD64-Rechner) einfügen. Nachdem Sie vorhin distcc emerged haben, sieht das Verzeichnis /usr/lib/distcc/bin wie folgt aus:

Die folgenden Anweisungen dürfen nur auf dem Rechner ausgeführt werden, der den emerge-Prozess ausführt. Führen Sie diese Schritte nicht auf den Hilfsrechnern aus.

Weitere Informationen

- Quelle Cross-Compiling