

1 Installation von BitKeeper

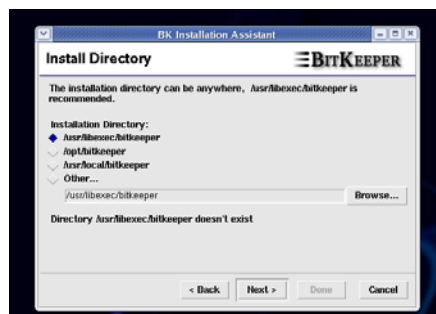
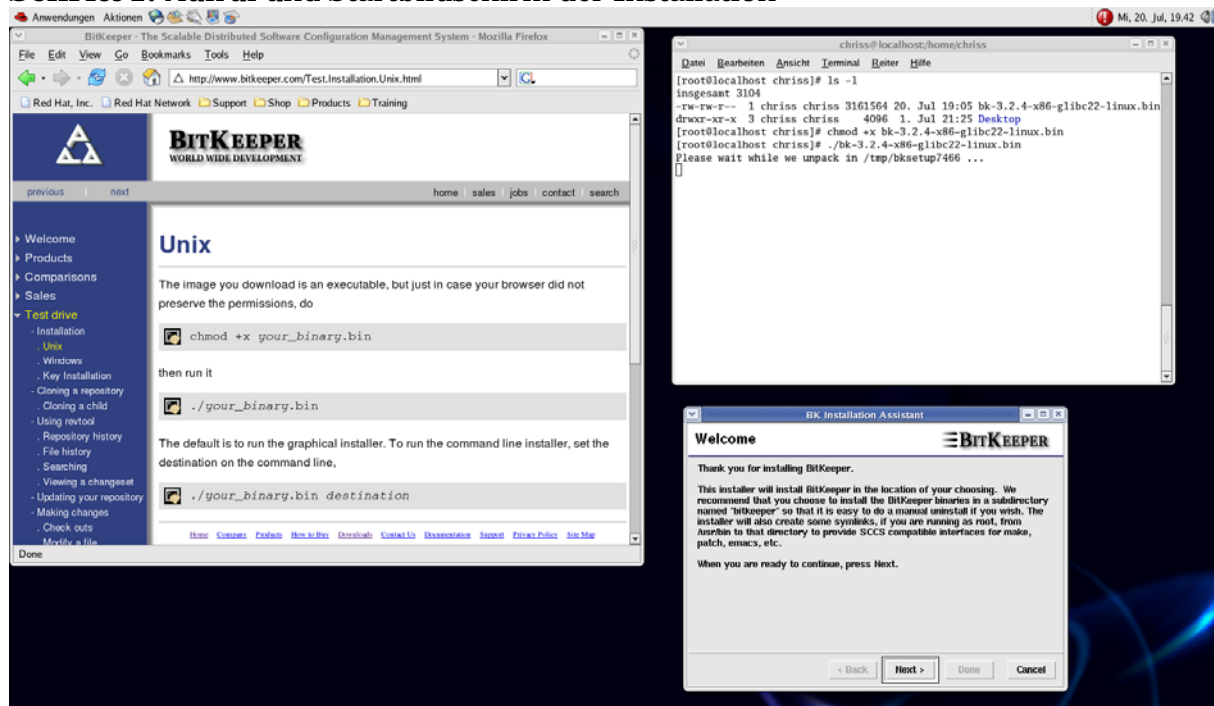
1.1 Download

BitKeeper kann von der Seite www.bitkeeper.com unter Products -> BK/Pro-> Downloads kostenlos heruntergeladen werden. Seit 1. Juli 2005 braucht BitKeeper allerdings einen Kommerziellen- oder einen Evaluations-Lizenzschlüssel. Man muß sich zuerst auf der Homepage registrieren lassen und bekommt dann erst den Verweis zum Download der Installationsdateien und die Lizenz.

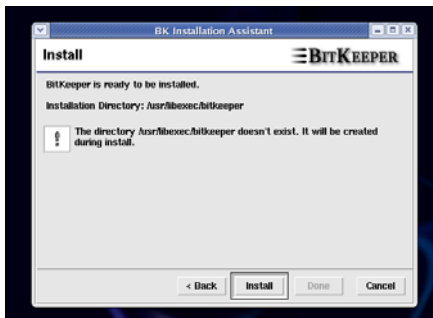
1.2 Installation unter Linux

Nach dem Download Bitkeeper einfach mit folgendem Befehl installieren:
./my_binary.bin

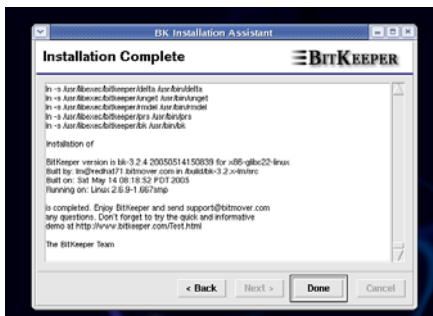
Schritt 1: Aufruf und Startbildschirm der Installation



Schritt 2: Zielverzeichnis auswählen.



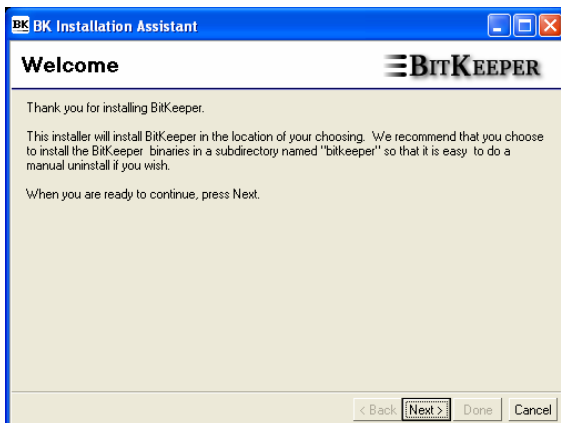
Schritt 3: Bei Installation ohne Root-Rechte kommt zusätzlich die Fehlermeldung, daß auf bestimmte Verzeichnisse nicht zugegriffen werden kann.



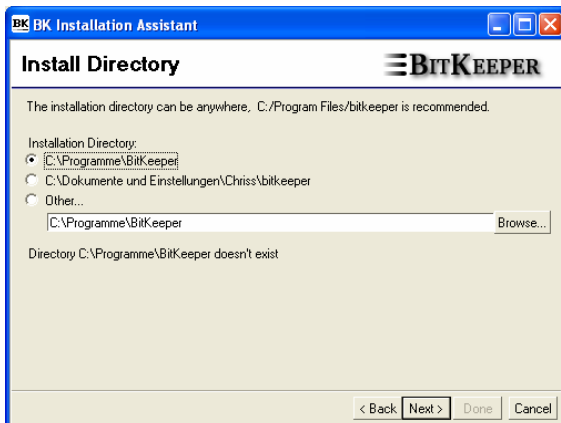
Schritt 4: Das Produkt wurde erfolgreich installiert.

1.3 Installation unter Windows

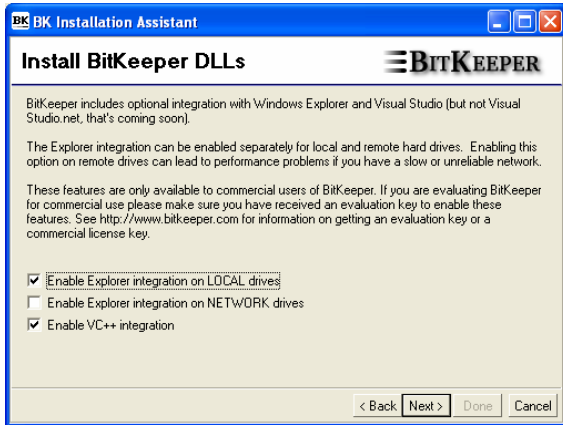
Nach dem Download BitKeeper einfach die Setup-Datei starten:



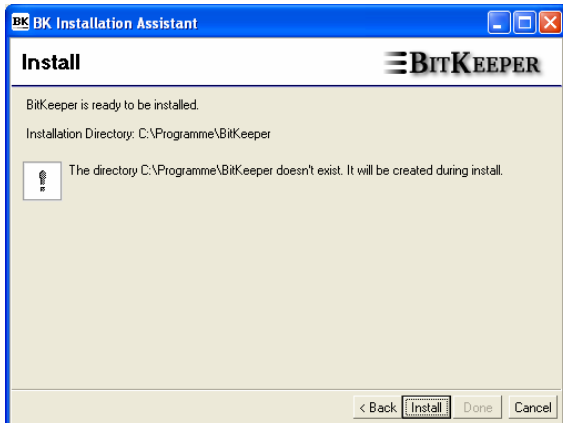
Schritt 1: Startbildschirm



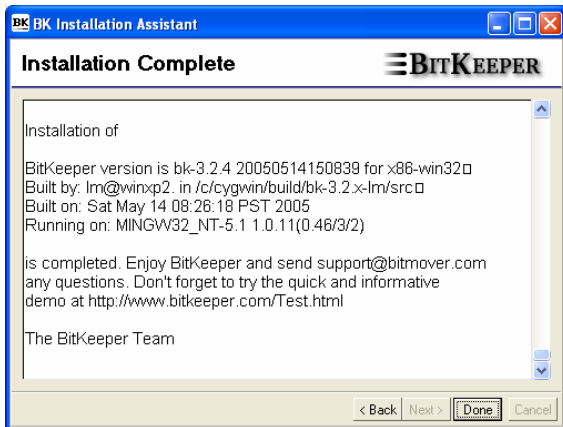
Schritt 2: Verzeichnis auswählen



Schritt 3: Shell Erweiterungen und Visual Studio Unterstützung einstellen. Überraschenderweise unterstützt BitKeeper immer noch KEIN Visual Studio .NET. Für ein kommerzielles Produkt dieser Größe eher enttäuschend, da Visual Studio .NET zumindest unter Windows de facto Standard ist.



Schritt 4: Hinweis über zu erzeugendes Verzeichnis.



Schritt 5: Installation erfolgreich abgeschlossen.

2 Arbeiten mit BitKeeper

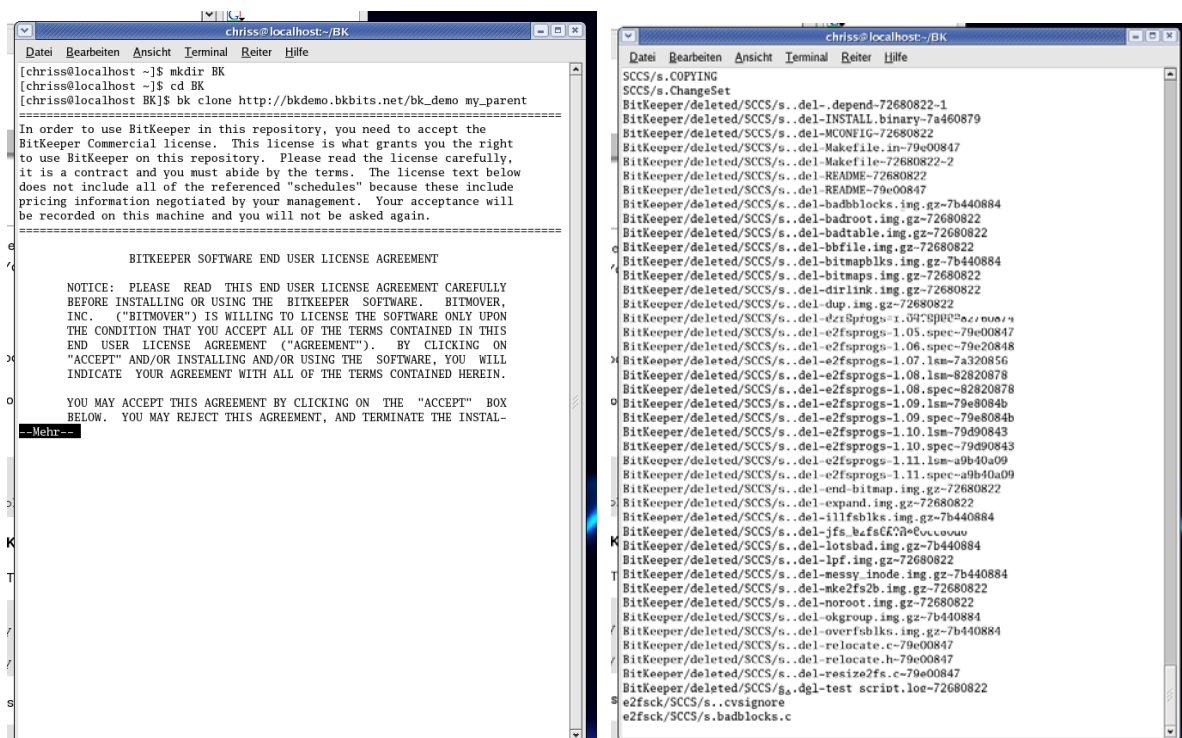
2.1 Erzeugen / Klonen von Repositories

Um seinen privaten Workspace zu bekommen muß man das globale Repository klonen:

```
bk clone <src_url> <name>
```

z.B.:

```
bk clone http://bkdemo.bkbits.net/bk_demo my_parent
```



Da man bei dem im Beispiel geladenem Demo-Repository `my_parent` keinen Schreibzugriff hat, muß man das eben erstellte Repository lokal noch einmal klonen:

Lokal Klonen:

```
bk clone <src> <dest>
```

z.B.:

```
bk clone my_parent bk_demo
```

Danach steht im Verzeichnis `bk_demo` eine funktionsfähige Kopie des Repositories zur Verfügung.

2.2 Update

Um seinen lokalen Workspace mit dem Repository abzugleichen (Update), gibt man folgenden Befehl ein:

```
bk pull
```

Dadurch wird mit dem Standard Repository abgeglichen. Man kann aber auch ein beliebiges, anderes Repository wählen:

```
z.B.:  
bk pull http://bkdemo.bkbits.net/bk_demo1
```

2.3 Check Out

Nach einem Update sind alle Dateien read-only. Das bedeutet, daß man sie zwar lesen kann, aber keine Änderungen im Quellcode vornehmen darf. Wenn man eine Datei, oder ein Verzeichnis bearbeiten will, wechselt man in dieses Verzeichnis und gibt folgendes ein:

```
bk edit
```

oder für einzelne Dateien:

```
bk edit <filename>
```

```
z.B.:  
bk edit mysource.cpp
```

Danach kann diese Datei bearbeitet werden.

2.5 Änderungen anzeigen

Um alle Veränderungen zu sehen genügt folgender Aufruf:

```
bk diff
```

2.4 Neue Datei einfügen

Falls eine neue Datei angelegt werden soll kann man diese einfach im passenden Verzeichnis erstellen.

Um zu sehen, welche Dateien noch nicht verwaltet sind (also neu angelegte) genügt dieser Befehl:

```
bk extras
```

Um die neu erstellte Datei mit in das Repository aufzunehmen wird der *new* Befehl verwendet:

```
bk new <filename>
```

z.B.:

```
bk new thenewfile.cpp
```

2.5 Check In

Falls alle Änderungen fertig sind und das Projekt lauffähig ist, sollte man es wieder einchecken, damit die Änderungen für jeden sichtbar werden. Anders als bei Versionsverwaltungen wie CVS, sind die Änderungen hier atomar. Das heißt erst beim finalen *Commit* werden alle Änderungen auch erst wirklich übernommen.

Eine veränderte Datei wieder einchecken:

```
bk delta -y"<comment>" <filename>
```

z.B.:

```
bk delta -y"Fehler 12 beseitigt" mysource.cpp
```

Um alle Änderungen sichtbar zu machen, muß die Transaktion abgeschlossen werden:

```
bk commit -y"<comment>"
```

z.B.:

```
bk commit -y"Modul xy eingefügt."
```

2.5 Merge

Falls andere Programmierer, in der Zwischenzeit, auch Änderungen an den vom Benutzer ausgecheckten Dateien vorgenommen haben, tritt ein Konflikt auf. Dieser wird beim Update-Versuch angezeigt (Pull):

```
bk pull
```

Ergebnis:

```
...
Conflicts during automerge of mysource.cpp
resolve: 1 unresolved conflicts, nothing is applied.
...
```

Jetzt muß man diesen Konflikt manuell beheben:

```
bk resolve
```

Dieser Konflikt-Löser hat ein paar Standard-Plugins, um alltägliche Konflikte (wie Verschieben, Einfügen, Löschen und Umbenennen von Code) automatisch zu lösen.

2.5 Vater-Repository aktualisieren

Bis jetzt wurden die Änderungen nur im lokalen Repository ausgeführt. Diese müssen noch in das Vater-Repository übernommen werden:

```
bk parent
bk push
```

3 Quellen

www.bikeeper.com