Handbok för skriptet kdesrc-build

Michael Pyne
Carlos Woelz
Översättare: Stefan Asserhäll

Handbok för skriptet kdesrc-build

Innehåll

Tabeller

Sammanfattning

kdesrc-build är ett skript som bygger och installerar KDEs programvara, direkt från källkoden som hämtas från .

Kapitel 1

Inledning

1.1 En kortfattad introduktion till kdesrc-build

1.1.1 Vad är kdesrc-build?

kdesrc-build är ett skript för att hjälpa KDE-gemenskapen installer KDE-programvara från Git källkodsarkiv, och fortsätta att uppdatera programvaran efteråt. Det är i synnerhet avsett att stödja de som behöver hantera testning och utveckling av KDE-programvara, inklusive användare som testar felrättningar och utvecklare som arbetar på nya funktioner.

Skriptet kdesrc-build kan användas för att underhålla en enskild individuell modul, ett fullständigt Plasma-skrivbord med KDE:s programuppsättning, eller någonting däremellan.

Se kapitel 2 för att komma igång, eller fortsätt läsa för mer detaljerad information om hur kdesrc-build fungerar och vad som omfattas av den här dokumentationen.

1.1.2 Funktionen hos kdesrc-build‘i ett nötskal’

kdesrc-build fungerar genom att använda verktygen tillgängliga för användaren på kommandoraden, och använder samma gränssnitt som är tillgängliga för användaren. När kdesrc-build kör, utförs följande sekvens:

1. kdesrc-build läser in kommandoraden och en inställningsfil för att bestämma vad som ska byggas, var det ska installeras, etc.
2. kdesrc-build utför en källkodsuppdatering för varje modul. Uppdateringen fortsätter tills alla moduler har uppdaterats. Moduler vars uppdatering misslyckas stoppar normalt inte bygget: du blir informerad i slutet om vilka moduler som inte uppdaterades.
3. Moduler som uppdaterades med lyckat resultat byggs, deras testsviter körs, och installeras därefter. För att reducera den totala tiden som går åt, börjar kdesrc-build normalt bygga koden så snart uppdateringen av den första modulen är klar, och låta återstående uppdateringar fortsätta bakom kulisserna.

TIPS

En *mycket bra* översikt av hur KDE-moduler byggs, inklusive informativa diagram, är tillgänglig i en artikel på nätet som beskriver KDE-programmet Krita. Det här arbetsflödet är vad kdesrc-build automatiserar för alla KDE-moduler.

1.2 Översikt över dokumentationen

Den här guiden är en översikt som beskriver följande aspekter vid användning av kdesrc-build:

- En översikt av stegen som krävs för att komma igång.
- Funktioner värda att lägga märke till.
- Inställningsfilens syntax och alternativ.
- Kommandoradsväljarna.

Dessutom dokumenteras stegen du måste utföra med andra verktyg (dvs. steg som inte utförs automatiskt av kdesrc-build).

Kapitel 2

Komma igång

I det här kapitlet visar vi hur kdesrc-build används för att checka ut moduler från KDE-arkivet och bygga dem. Vi tillhandahåller också en grundläggande förklaring av KDE:s källkodsstruktur och stegen du måste utföra innan skriptet körs.

Alla ämnen som presenteras i det här kapitlet täcks med ännu mer detaljer i artikeln Build from Source på webbplatsen KDE Community Wiki. Om du kompilerar KDE för första gången, är det en god idé att läsa den, eller rådfråga den som en referenskälla. Du hittar detaljerad information om paketverktyg och krav, vanliga fallgropar vid kompilering och strategier och information om att köra den nya KDE-installationen.

2.1 Förbereda systemet för att bygga KDE

2.1.1 Ställa in ett nytt användarkonto

Det rekommenderas att du använder ett annat användarkonto för att bygga, installera och köra din KDE-programvara från, eftersom färre rättigheter krävs, och för att undvika konflikt med distributionens paket. Om du redan har installerade KDE-paket, är det bästa valet att skapa en annan (dedicerad) användare för att bygga och köra det nya KDE.

> TIPS
> Att lämna systemets KDE orört, låter dig också ha en reservutväg i nödfall om ett kodningsmisstag gör att den senaste programvarubyggningen är oanvändbar.

Du kan också ställa in att installera i en systemkatalog (t.ex. `/usr/src/local`) om du vill. Det här dokumentet omfattar inte den installationstypen, eftersom vi antar att du vet vad du gör.

2.1.2 Försäkra dig om att systemet är klart att bygga KDE:s källkod

Innan du använder skriptet kdesrc-build (eller någon annan byggstrategi) måste du installera utvecklingsverktyg och bibliotek som behövs för KDE. Den nästan fullständiga listan med verktyg som behövs finns på sidan med byggkrav på KDE Community Wiki .

Här är en lista med några av de saker du kommer att behöva:

- Du behöver CMake, eftersom denna programvara är vad KDE använder för att hantera byggkonfiguration av källkoden och generering av de specifika byggkommandona för systemet. Versionen som krävs varierar beroende på vilken version av KDE-programvaran som byggs (se teknikbasen för närmare detaljer), men med moderna distributioner bör den CMake som är inkluderad i distributionen vara fullt tillräcklig.
- Du måste också installera klientprogram för källkodshantering som används för att checka ut KDE:s källkod. Det betyder att du åtminstone behöver följande:
  - Källkodshanteringssystemet Git som används för all källkod i KDE.
  - Även om det inte krävs, används källkodshanteringssystemet Bazaar för en enda modul (libdbusmenu-qt) som krävs av KDE-biblioteken. De flesta användare kan installera biblioteket via distributionens paket, men kdesrc-build stöder att också bygga det om du så önskar. Men för att bygga libdbusmenu-qt, måste du ha installerat Bazaar.
- Skriptspråket Perl krävs för kdesrc-build, vissa KDE-arkiv och Qt™ (om det byggs från källkod).

  Perl som levereras med distributionen bör vara lämplig (det måste vara minst Perl 5.14), men vissa ytterligare moduler behövs också (kdesrc-build varnar dig om de inte är tillgängliga):
  - IO::Socket::SSL
  - JSON::PP eller JSON::XS
  - YAML::PP, YAML::XS eller YAML::Syck
- En fullständig C++ utvecklingsmiljö behövs (kompilator, standardbibliotek, körtidsbibliotek och alla utvecklingspaket som krävs). De äldsta nödvändiga versionerna varierar baserat på KDE-modulen: samlingen KDE-ramverk 5 stöder de äldsta kompilatorerna, medan KDE Plasma 5 och KDE-program brukar kräva nyare kompilatorer.

  Kompilatorerna GCC 4.8 eller Clang 4 är de äldsta som rekommenderas. Många distributioner stöder att enkelt installera verktygen genom att använda paketet 'build-essentials', ett alternativ att installera "byggberoenden" Qt™ eller liknande funktioner. KDE-gemenskapens Wiki har en sida som följer rekommenderade paket för större distributioner.
- Ett byggverktyg som faktiskt utför kompileringsstegen behövs (de som genereras av CMake). GNU Make rekommenderas och bör vara tillgängligt via pakethanteraren. CMake stöder andra alternativ, såsom byggverktyget Ninja, som kan utnyttjas av kdesrc-build genom att använda inställningsfilens alternativ custom-build-command.
- Slutligen behövs lämpliga Qt™-bibliotek (inklusive utvecklingspaket) för versionen av KDE-programvara som du bygger. kdesrc-build stöder inte officiellt att bygga Qt™ 5 (nuvarande huvudversion), alltså rekommenderas du att använda distributionens utvecklingspaket, eller att titta på KDE-gemenskapens Wiki sida om att bygga Qt 5 själv.

> **NOT**
> De flesta distributioner av operativsystem inkluderar en metod att enkelt installera nödvändiga utvecklingsverktyg. Konsultera avsnittet Required devel packages på Community Wiki, för att se om instruktioner redan är tillgängliga.

> **VIKTIGT**
> Vissa av paketen är uppdelade i bibliotek (eller program, eller verktyg) och utvecklingspaket Du behöver åtminstone programmet eller biblioteket *och* dess utvecklingspaket.

2.1.3 Inställning av kdesrc-build

2.1.3.1 Installera kdesrc-build

KDE-utvecklarna gör täta förändringar av kdesrc-build för att hålla det synkroniserat med framsteg i KDE-utvecklingen, inklusive förbättringar av den rekommenderade inställningen av kdesrc-build, tillägg av moduler, förbättring av flaggor i CMake, etc.

På grund av detta, rekommenderar vi att hämta kdesrc-build direkt från dess källkodsarkiv och därefter uppdatera det periodiskt.

Du kan hämta kdesrc-build från dess källkodsarkiv, genom att köra:

```
$ git clone https://invent.kde.org/sdk/kdesrc-build.git ~/kdesrc-build
```

Ersätt `~/kdesrc-build` med katalogen du vill installera i.

Du kan senare uppdatera kdesrc-build genom att köra:

```
$ cd ~/kdesrc-build
$ git pull
```

> **TIPS**
> Vi rekommenderar att lägga till installationskatalogen för kdesrc-build i miljövariabeln `PATH`, så att kdesrc-build kan köras utan att behöva ange hela sökvägen varje gång.

2.1.3.2 Färdigställa inställningsfilen

kdesrc-build använder en inställningsfil för att bestämma vilka moduler som byggs, var de installeras, etc. Filen heter `~/.config/kdesrc-buildrc` (`$XDG_CONFIG_HOME/kdesrc-buildrc`, om `$XDG_CONFIG_HOME` är angivet).

Du kan använda ett program som ingår tillsammans med kdesrc-build, vid namn kdesrc-build-setup för att skapa en enkel inställning för kdesrc-build. Därefter kan du redigera inställningsfilen `~/.config/kdesrc-buildrc` för att göra eventuella ändringar du har behov av.

Själva kdesrc-build-setup körs från en terminal (istället för att använda ett grafiskt gränssnitt), precis som kdesrc-build, så du kan använda det även om du inte har något grafiskt gränssnitt tillgängligt ännu.

2.1.3.2.1 Manuell uppdatering av inställningsfilen

Det går också att uppdatera inställningsfilen för hand, genom att kopiera det inkluderade exemplet på en inställningsfil `kdesrc-buildrc-kf5-sample` till `~/.config/kdesrc-buildrc` och sedan redigera filen. En användbar referens för detta är kapitel 4, i synnerhet dess tabell över inställningsalternativ.

kdesrc-build innehåller många rekommenderade inställningsfiler för att stödja KDE Ramverk 5, Plasma 5 och andra KDE-program. kdesrc-build-setup hänvisar till dessa filer i inställningsfilen som skapas, men du kan också använda dem själv. Se Avsnitt 4.1.2 för information om hur andra inställningsfiler används från din egen `kdesrc-buildrc`.

Du hittar mer information om syntaxen i en inställningsfil i Avsnitt 2.2 och kapitel 4.

2.2 Ange inställningsinformation

För att använda kdesrc-build ska du ha en fil i katalogen `~/.config` (eller i `$XDG_CONFIG_HOME`, om angivet) vid namn `kdesrc-buildrc`, som ställer in allmänna alternativ och anger modulerna som du vill ladda ner och bygga.

> **NOT**
> Det är möjligt att använda andra inställningsfiler för kdesrc-build, som beskrivs i kapitel 4. Om du behöver använda flera inställningar, se det avsnittet.Här antar vi att inställningarna är lagrade i `~/.config/kdesrc-buildrc`.

Det enklaste sättet att fortsätta är att använda filen `kdesrc-buildrc-kf5-sample` som mall, och ändra allmänna inställningar för att stämma med vad du vill ha, och dessutom ändra listan med moduler du vill bygga.

Standardinställningarna bör vara lämpliga för att utföra byggprocessen för KDE. Vissa inställningar som du kan vilja ändra omfattar:

- kdedir, som ändrar målkatalogen som KDE-programvaran installeras i. Standardvärdet är `~/kde`, som är en enanvändarinstallation.
- branch-group som kan användas för att välja lämplig utvecklingsgren för KDE-moduler som helhet. Det finns många byggkonfigurationer som stöds, men det är troligtvis `kf5-qt5` som du ska välja så att kdesrc-build laddar ner den senaste koden baserad på Qt™ 5 och KDE Ramverk 5.

  > **TIPS**
  > kdesrc-build använder en standardgrupp om du inte väljer någon, men standardvärdet ändras med tiden, så det är bättre att välja en så att inte grengruppen oväntat ändras.

- source-dir, för att bestämma katalogen som kdesrc-build använder för att ladda ner källkoden, köra byggprocessen och spara loggar. Standardvärdet är `~/kdesrc`.
- cmake-options som ställer in alternativ att skicka till kommandot CMake när varje modul byggs. Oftast används det för att välja mellan byggvarianterna ‘debug’ och ‘release’, för att aktivera (eller inaktivera) valfria funktioner, eller för att skicka information till byggprocessen om platsen för nödvändiga bibliotek.
- make-options som ställer in väljare använda när kommandot make faktiskt utförs för att bygga varje modul (när väl CMake har etablerat byggsystemet).

  Den mest typiska väljaren är `-jN`, där *N* ska ersättas med det maximala antal kompileringsjobb som du vill tillåta. Ett större tal (upp till antalet logiska processorer som systemet har tillgängligt) leder till en snabbare byggprocess, men kräver mer systemresurser.

  > **TIPS**
  > kdesrc-build ställer in alternativet `num-cores` till detekterat antal processorkärnor. Du kan använda värdet i din egen inställningsfil för att undvika att behöva ange det manuellt.

Example 2.1 Anpassa Make för att använda alla tillgängliga processorer, med undantag

```
global
    # Miljövariabeln används automatiskt av make, inklusive
    # make-kommandon som inte direkt körs av kdesrc-build, såsom Qt:s ↩
       configure
    set-env MAKEFLAGS -j${num-cores}
    &#8230;
end global

&#8230;

module-set big-module-set
    repository kde-projects
    use-modules calligra
    make-options -j2 # Reducerat antal byggjobb för bara dessa moduler
end module-set
```

> NOT
> Några mycket stora Git-arkiv kan översvämma systemet om du försöker kompilera med för många byggjobb på en gång, särskilt arkiv som Qt™ Webkit och Qt™ WebEngine. För att behålla systemets interaktivitet måste du kanske reducera antal byggjobb för specifika moduler.
> Exempel 2.1 ger ett exempel på hur man kan göra det.

Du kanske vill välja andra moduler att bygga, vilket beskrivs i Avsnitt 2.6.2.

2.3 Att använda skriptet kdesrc-build

När inställningsdata är upprättad, är du redo att köra skriptet. Även om du fortfarande har en del finjustering eller annat som du vill läsa, är det en god idé att åtminstone läsa in KDE-projektets metadata.

2.3.1 Läsa in projektmetadata

Logga in till användaren som du använder för att kompilera KDE:s programvara i ett terminalfönster, och kör skriptet:

```
% kdesrc-build --metadata-only
```

Kommandot ställer in källkodskatalogen och ansluter till KDE:s Git-arkiv för att ladda ner databasen med KDE:s git-arkiv, och databasen med metadata över beroenden, utan att göra några ytterligare ändringar. Det är användbart att göra det separat, eftersom metadata är användbar för andra kommandon i kdesrc-build.

2.3.2 Förhandsgranska vad som kommer att hända när kdesrc-build kör

När projektets metadata är installerad är det möjligt att granska vad kdesrc-build kommer att göra när det startas. Det kan åstadkommas med kommandoradsväljaren `--pretend`.

```
% ./kdesrc-build --pretend
```

Du ska se ett meddelande som talar om att några paket har byggts med lyckat resultat (även om ingenting faktiskt byggdes). Om inga väsentliga problem visas, kan du fortsätta att verkligen köra skriptet.

```
% kdesrc-build
```

Kommandot laddar ner lämplig källkod, bygger och installerar varje modul i tur och ordning. Efteråt ska du se utmatning som liknar den i Exempel 2.2.

Example 2.2 Exempel på utmatning från en kdesrc-build körning

```
% kdesrc-build
Updating kde-build-metadata (to branch master)
Updating sysadmin-repo-metadata (to branch master)

Building libdbusmenu-qt (1/200)
        No changes to libdbusmenu-qt source, proceeding to build.
        Compiling... succeeded (after 0 seconds)
        Installing.. succeeded (after 0 seconds)

Building taglib (2/200)
        Updating taglib (to branch master)
        Source update complete for taglib: 68 files affected.
        Compiling... succeeded (after 0 seconds)
        Installing.. succeeded (after 0 seconds)

Building extra-cmake-modules from <module-set at line 32> (3/200)
        Updating extra-cmake-modules (to branch master)
        Source update complete for extra-cmake-modules: 2 files affected.
        Compiling... succeeded (after 0 seconds)
        Installing.. succeeded (after 0 seconds)

        ...

Building kdevelop from kdev (200/200)
        Updating kdevelop (to branch master)
        Source update complete for kdevelop: 29 files affected.
        Compiling... succeeded (after 1 minute, and 34 seconds)
        Installing.. succeeded (after 2 seconds)

<<<  PACKAGES SUCCESSFULLY BUILT  >>>
Built 200 modules

Your logs are saved in /home/kde-src/kdesrc/log/2018-01-20-07
```

2.3.3 Lösa byggfel

Beroende på hur många moduler du laddar ner, är det möjligt att kdesrc-build inte lyckas första gången du kompilerar KDE:s programvara. Ge inte upp hoppet!

kdesrc-build loggar utmatningen från varje kommando som körs. Normalt lagras loggfilerna i `~/kdesrc/log`. För att se vad som orsakade ett fel för en modul vid det senaste kdesrc-build-kommandot, är det oftast tillräckligt att titta i `~/kdesrc/log/latest/ modulnamn /error.log`.

> **TIPS**
> Det kanske enklaste sättet att ta reda på vilka fel som gör att bygget av en modul misslyckas är att söka bakåt efter ordet `error` från slutet av filen med en sökning som inte är skiftlägeskänslig. När det hittats, rulla uppåt för att vara säker på att det inte finns några andra felmeddelanden i närheten. Det första felmeddelandet i en grupp är ofta det bakomliggande problemet.

I filen ser du felet som orsakade att byggprocessen misslyckades för modulen. Om det står (längst ner) i filen att vissa paket saknas, försök att installera paketet (inklusive eventuella lämpliga -dev paket) innan du försöker bygga modulen igen. Försäkra dig om att skicka med väljaren --reconfigure när kdesrc-build körs igen, så att kdesrc-build gör att modulen letar efter de saknade paketen igen.

Eller om felet verkar vara ett byggfel (som ett syntaxfel, 'incorrect prototype', 'unknown type' eller likande) är det nog ett fel i KDE:s källkod, som förhoppningsvis löses inom några dagar. Om det inte har lösts inom den tiden, skicka gärna ett brev till e-postlistan kde-devel@kde.org (prenumeration kan krävas först) för att rapportera byggfelet.

Du hittar fler vanliga exempel på saker som kan gå fel och deras lösningar, samt allmänna tips och strategier för att bygga KDE:s programvara i Build from Source.

Å andra sidan, under förutsättning att allt gick bra, ska en ny KDE-version vara installerad på datorn, och nu är det helt enkelt bara en fråga om att köra den, vilket beskrivs härnäst i Avsnitt 2.5.

> **NOT**
> För mer information om loggningssystemet i kdesrc-build, se Avsnitt 3.2.

2.4 Bygga specifika moduler

Istället för att bygga alla moduler hela tiden, kanske man vill bygga en enstaka modul, eller någon annan liten delmängd. Istället för att redigera inställningsfilen, kan man helt enkelt skicka med namn på moduler eller moduluppsättningar att bygga på kommandoraden.

Example 2.3 Exempel på utmatning när en specifik modul byggs av kdesrc-build

```
% kdesrc-build --include-dependencies dolphin
Updating kde-build-metadata (to branch master)
Updating sysadmin-repo-metadata (to branch master)

Building extra-cmake-modules from frameworks-set (1/79)
        Updating extra-cmake-modules (to branch master)
        No changes to extra-cmake-modules source, proceeding to build.
        Running cmake...
        Compiling... succeeded (after 0 seconds)
        Installing.. succeeded (after 0 seconds)

Building phonon from phonon (2/79)
        Updating phonon (to branch master)
        No changes to phonon source, proceeding to build.
        Compiling... succeeded (after 0 seconds)
        Installing.. succeeded (after 0 seconds)

Building attica from frameworks-set (3/79)
        Updating attica (to branch master)
        No changes to attica source, proceeding to build.
        Compiling... succeeded (after 0 seconds)
        Installing.. succeeded (after 0 seconds)

        ...

Building dolphin from base-apps (79/79)
        Updating dolphin (to branch master)
        No changes to dolphin source, proceeding to build.
        Compiling... succeeded (after 0 seconds)
        Installing.. succeeded (after 0 seconds)

<<<  PACKAGES SUCCESSFULLY BUILT  >>>
Built 79 modules

Your logs are saved in /home/kde-src/kdesrc/log/2018-01-20-07
```

Även om bara programmet *dolphin* angavs i detta fall, orsakade väljaren `--include-dependencies` att kdesrc-build inkluderade beroenden listade för *dolphin* (genom att ange väljaren include-dependencies).

> NOT
> I detta fall fungerade beroendehanteringen bara eftersom *dolphin* råkar vara angiven i en moduluppsättning baserad på `kde-projects` (kallad `base-apps` i exemplet). Se Avsnitt 2.6.3.2.

2.5 Ställa in miljön för att köra ditt KDEPlasma-skrivbord

Med antagandet att du använder en särskild användare för att bygga KDE Plasma och redan har en installerad version av Plasma, kan det vara något krångligt att köra det nya Plasma, eftersom det nya måste ges företräde över det gamla. Du måste ändra miljövariabler i inloggningsskript för att försäkra dig om att det nyss byggda skrivbordet används.

2.5.1 Installerar automatiskt en drivrutin för inloggning

Från och med version 1.16 försöker kdesrc-build installera en lämplig drivrutin för inloggning, som låter dig logga in på KDE-skrivbordet som byggts av kdesrc-build från din inloggningshanterare. Det kan inaktiveras genom att använda inställningsalternativet `install-session-driver` i inställningsfilen.

> NOT
> Sessionsinställning sker inte medan kdesrc-build gör i låtsasläge.

Drivrutinen fungerar genom att skapa en egen ‘`xsession`’ sessionstyp. Denna typ av session ska fungera direkt med inloggningshanteraren sddm (där den visas som en ‘Egen’ session), men andra inloggningshanterare (som LightDM och gdm) kan kräva att ytterligare filer installeras för att aktivera stöd för `xsession`.

2.5.1.1 Lägger till xsession-stöd för distributioner

Den förinställda inloggningshanteraren för vissa distributioner kan kräva att ytterligare paket installeras för att stödja inloggning med `xsession`.

- Distributionen Fedora Linux® kräver att paketet `xorg-x11-xinit-session` är installerat för att stödja inloggning med en egen `xsession`.
- Debian och Linux®-distributioner baserade på Debian ska stödja inloggning med en egen `xsession`, men kräver att alternativet `allow-user-xsession` ställs in i `/etc/X11/Xsession.options`. Se också Debians dokumentation om att anpassa en X-session.
- För övriga distributioner, se Avsnitt 2.5.1.2.

2.5.1.2 Manuellt tillägg av stöd för xsession

Om det inte fanns några distributionsspecifika anvisningar för din distribution i Avsnitt 2.5.1.1, kan du lägga till en post i distributionens lista med sessionstyper för ‘inloggning med egen xsession’ på följande sätt:

> NOT
> Proceduren kräver troligen administratörsrättigheter för att kunna göras färdig.

1. Skapa filen `/usr/share/xsessions/kdesrc-build.desktop`.
2. Försäkra dig om att filen som just skapas innehåller följande text:

```
Type=XSession
Exec=$HOME/.xsession
Name=KDE Plasma Desktop (unstable; kdesrc-build)
```

   ❶ Värdet *`$HOME`* måste ersättas av den fullständiga sökvägen till din hemkatalog (exempelvis `/home/ användare`). Specifikationen av skrivbordsposter tillåter inte generella användarfiler.

3. När inloggningshanteraren startas om, ska den visa en ny sessionstyp, ‘KDE Plasma Desktop (unstable; kdesrc-build)’ i listan med sessioner, vilket ska försöka köra filen `.xsession` som installerats av kdesrc-build, om den är markerad när du loggar in.

> NOT
> Det kan vara enklast att starta om datorn för att starta om inloggningshanteraren, om inloggningshanteraren inte bevakar uppdateringar av katalogen `/usr/share/xsessions`.

2.5.2 Ange miljön för hand

Dokumentationen innehöll tidigare instruktioner om vilka miljövariabler som ska ställas in för att kunna läsa in det nybyggda skrivbordet. Dessa instruktioner har flyttats till ett appendix (Avsnitt B.1).

Om du har för avsikt att ställa in eget stöd för inloggning, kan du titta i appendixet eller öppna filen `sample-kde-env-master.sh` som ingår i källkoden för kdesrc-build.

2.6 Organisation och urval av moduler

2.6.1 Organisation av KDE-programvara

KDE-programvara är uppdelad i olika komponenter, där en stor del kan byggas av kdesrc-build. Att förstå organisationen hjälper till att välja programvarumoduler att bygga.

1. På lägsta nivån finns Qt™-biblioteket, som är en mycket kraftfull, plattformsoberoende 'verktygslåda'. KDE är baserat på Qt™ och vissa av de bibliotek som inte hör till KDE, men krävs av KDE, är också baserade på Qt™. kdesrc-build kan bygga Qt™ eller använda ett som redan är installerat på systemet om versionen är tillräckligt ny.
2. Ovanpå Qt™ finns nödvändiga bibliotek som krävs för att KDE-programvara ska fungera. Vissa av dessa bibliotek anses inte vara en del av själva KDE, på grund av deras generella natur, med är ändå väsentliga för KDE-plattformen. Dessa bibliotek samlas i modulen`kdes upport`, men anses inte vara en del av 'Ramverk-biblioteken'.
3. Ovanpå de här väsentliga biblioteken kommer KDE Ramverk, ibland förkortat som KF5, vilka är väsentliga bibliotek för KDE:s Plasma-skrivbord, KDE-program och annan tredjepartsprogramvara.
4. Ovanpå ramverket, finns flera olika saker:
   - 'Tredjepartsprogram'. Detta är program som använder KDE:s ramverk eller är konstruerade för att köra med KDE Plasma, men inte skapas av eller i samröre med KDE-projektet.
   - Plasma, som är en komplett skrivbordsmiljö, 'arbetsyta'. Det är vad användare normalt ser när de 'loggar in med KDE'.
   - KDE:s programvarusvit. Det är en samling användbar programvara som inkluderas med plattformen och Plasma-skrivbord, grupperade i individuella moduler, inklusive verktyg som Dolphin, spel som Ksudoku och produktivitetsprogram som ges ut av KDE, såsom Kontact.
   - Till sist, finns en samling programvara (också samlad i moduler) vars utveckling stöds av resurser som tillhandahålls av KDE (såsom översättning, källkodskontroll, feluppföljning, etc.), men inte ges ut av KDE eller som en del av Plasma eller programvarusamlingen. Dessa moduler kallas för 'extragear'.

2.6.2 Välja moduler att bygga

Att välja vilka av möjliga moduler som ska byggas styrs av inställningsfilen. Efter sektionen `glo bal` finns en lista av moduler att bygga, omgivna av raderna module ... end module. Ett exempel på en post för en modul visas i Exempel 2.4.

Example 2.4 Exempel på modulpost i inställningsfilen

```
module kdesrc-build-git
    # Alternativ för modulen anges här, exempelvis:
    repository kde:kdesrc-build
    make-options -j4 # Kör 4 kompileringar samtidigt
end module
```

> NOT
> I praktiken används oftast inte modulkonstruktionen direkt. Istället anges de flesta moduler via moduluppsättningar som beskrivs nedan.

När bara poster som `module` används, bygger kdesrc-build dem i ordningen som anges, och försöker inte ladda ner några andra arkiv utom de som är direkt angivna.

2.6.3 Moduluppsättningar

KDE:s källkod är uppdelad i ett stort antal relativ små Git-baserade arkiv. För att göra det enklare att hantera det stora antalet arkiv som är inblandade i alla användbara KDE-baserade installationer, stöder kdesrc-build gruppering av flera moduler och att behandla gruppen som en 'moduluppsättning'.

2.6.3.1 Moduluppsättningarnas grundkoncept

Genom att använda en moduluppsättning kan man enkelt deklarera att många Git-moduler ska laddas ner och byggas, som om en separat moduldeklaration skrevs ut för var och en av dem. Alternativet repository hanteras på ett särskilt sätt för att ställa in varifrån varje modul laddas ner, medan alla andra alternativ som finns i moduluppsättningen kopieras till varje modul som skapas på detta sätt.

Example 2.5 Använda moduluppsättningar

```
global
    git-repository-base kde-git kde:
end global

module qt
    # Alternativ borttagna för korthets skull
end module

module-set kde-support-libs
    repository kde-git
    use-modules automoc attica akonadi
end module-set

# Övriga moduler efter behov...
module kdesupport
end module
```

I Exempel 2.5 visas en kortfattad moduluppsättning. När kdesrc-build träffar på moduluppsättningen, beter det sig som om en individuell modul har skapats för varje modul angiven i `use-modules` lika med moduluppsättningens `repository` omedelbart följt av det angivna modulnamnet.

Dessutom kan andra alternativ tas med i en moduluppsättning, som kopieras till varje ny modul som skapas på detta sätt. Genom att använda moduluppsättningar är det möjligt att snabbt deklarera många Git-moduler som är baserade på webbadressen för samma arkiv. Dessutom är det möjligt att namnge moduluppsättningar (som visas i exemplet), vilket gör det möjligt att snabbt referera till hela gruppen av moduler på kommandoraden.

2.6.3.2 Särskilt stöd för KDE:s moduluppsättningar

Stödet för moduluppsättningar som hittills beskrivits är generell för vilken Git-baserad modul som helst. För KDE:s Git-arkiv innehåller kdesrc-build ytterligare funktioner för att göra saker lättare för användare och utvecklare. Stödet aktiveras genom att ange `kde-projects` som `repository` för moduluppsättningen.

kdesrc-build bygger normalt bara modulerna som är listade i inställningsfilen, i ordningen som de listas. Men med moduluppsättningen `kde-projects`, kan kdesrc-build utföra beroendehantering för KDE-specifika moduler, och dessutom automatiskt inkludera moduler i bygget även om de bara specificeras indirekt.

Example 2.6 Använda kde-projects moduluppsättningar

```
# Lägger bara till en modul för juk (arkivet kde/kdemultimedia/juk)
module-set juk-set
    repository kde-projects
    use-modules juk
end module-set

# Lägger till alla moduler som finns i kde/multimedia/*, inklusive juk,
# men inga andra beroenden
module-set multimedia-set
    repository kde-projects
    use-modules kde/multimedia
end module-set

# Lägger till alla moduler som finns i kde/multimedia/*, och alla beroenden
# på kde-projects utanför kde/kdemultimedia
module-set multimedia-deps-set
    repository kde-projects
    use-modules kde/multimedia
    include-dependencies true
end module-set

# Alla moduler som skapas av dessa tre moduluppsättningar läggs automatiskt ↩
    i
# korrekt beroendeordning, oberoende av inställningen av include- ↩
   dependencies
```

> **TIPS**
> Konstruktionen med moduluppsättningen `kde-projects` är huvudmetoden för att ange vilka moduler som man vill bygga.

Alla moduluppsättningar använder alternativen repository och use-modules. Moduluppsättningarna `kde-projects` har ett fördefinierat värde på `repository`, men andra typer av moduluppsättningar använder också alternativet git-repository-base.

2.6.4 KDE:s officiella moduldatabas

KDE:s Git-arkiv tillåter att relaterade Git-moduler grupperas i samlingar av relaterade moduler (t.ex. kdegraphics). Git känner inte till sådana grupperingar, men kdesrc-build kan förstå grupperna genom att använda moduluppsättningar med alternativet `repository` inställt till ‘`kde-projects`’.

kdesrc-build förstår att arkivet `kde-projects` kräver särskild hantering, och justerar kompileringsprocessen därefter. Bland annat kommer kdesrc-build att:

- Ladda ner den senaste moduldatabasen från KDE:s git-arkiv.
- Försök att hitta en modul med det givna namnet i moduluppsättningens inställning i databasen, `use-modules`.
- För varje modul som hittas, slår kdesrc-build upp lämpligt arkiv i databasen, i själva verket baserat på inställningen branch-group. Om ett arkiv finns och är aktivt för grengruppen, använder kdesrc-build det automatiskt för att ladda ner eller uppdatera källkoden.

> **NOT**
> I den nuvarande databasen har vissa modulgrupper inte bara en samling moduler, utan de anger *också* sitt eget Git-arkiv. I dessa situationer föredrar kdesrc-build för närvarande gruppens Git-arkiv istället för att inkludera delmodulernas arkiv.

Nästa exempel visar hur man använder KDEs moduldatabas för att installera multimediabiblioteket Phonon.

```
module-set media-support
    # Denna väljare måste vara kde-projects för att använda moduldatabasen.
    repository kde-projects

    # Denna väljare anger vilken modul som ska sökas efter i databasen.
    use-modules phonon/phonon phonon-gstreamer phonon-vlc
end module-set
```

> **TIPS**
> Här används `phonon/phonon` eftersom kdesrc-build annars skulle behöva välja mellan projektgruppen vid namn ‘phonon’ eller det enskilda projektet vid namn ‘phonon’ (med den nuvarande projektdatabasen). För närvarande skulle kdesrc-build välja det första, vilket skulle bygga många fler gränssnitt än vad som behövs.

Nästa exempel är kanske mer realistiskt, och visar en funktion som bara är tillgänglig med KDEs moduldatabas: Att bygga alla KDE:s grafikprogram med en enda deklaration.

```
module-set kdegraphics
    # Denna väljare måste vara kde-projects för att använda moduldatabasen.
    repository kde-projects

    # Denna väljare anger vilka moduler som ska sökas efter i databasen.
    use-modules kdegraphics/libs kdegraphics/*
end module-set
```

Här åskådliggörs två viktiga möjligheter:

1. kdesrc-build låter dig ange moduler som är härstammar från en given modul, utan att bygga modulen själv, genom att använda syntaxen **`modulnamn /*`**. Det krävs i själva verket i det här fallet, eftersom basmodulen, kdegraphics, är markerad som inaktiv, så att den inte av misstag byggs med sina ättlingar. Att ange ättlingar låter kdesrc-build hoppa förbi den inaktiverade modulen.
2. kdesrc-build lägger inte heller till en given modul i bygglistan mer än en gång. Det låter oss manuellt ange att `kdegraphics/libs` ska byggas först, innan resten av `kdegraphics` utan att försöka bygga `kdegraphics/libs` två gånger. Det krävdes tidigare för riktig beroendehantering, och är idag ett reservalternativ i fall KDE:s projektdatabas saknar metadata för beroenden.

2.6.5 Filtrera bort KDE:s projektmoduler

Du kan bestämma dig för att du vill bygga alla program i en gruppering av KDE-moduler *utom* ett visst program.

Exempelvis inkluderar gruppen `kdeutils` ett program som heter kremotecontrol. Om datorn inte har lämplig hårdvara för att ta emot signaler som skickas från fjärrkontroller kanske du bestämmer att du inte vill ladda ner, bygga och installera kremotecontrol varje gång som `kdeutils` uppdateras.

Du kan åstadkomma det genom att använda inställningsalternativet ignore-modules. På kommandoraden gör väljaren --ignore-modules samma sak, men är bekvämare för att bara filtrera bort en modul en enstaka gång.

Example 2.7 Exempel på att ignorera en kde-project modul i en grupp

```
module-set utils
    repository kde-projects

     # Denna väljare anger vilka moduler som ska sökas efter i databasen.
    use-modules kdeutils

    # Denna väljare "subtraherar bort" moduler från modulerna som väljes av ↩
        use-modules ovan.
    ignore-modules kremotecontrol
end module-set

module-set graphics
    repository kde-projects

    # Denna väljare anger vilka moduler som ska sökas efter i databasen.
    use-modules extragear/graphics

    #  Denna väljare "subtraherar bort" moduler från modulerna som väljes  ↩
       av use-modules ovan.
    # I detta fall, ignoreras *både* extragear/graphics/kipi-plugins
    # och extragear/graphics/kipi-plugins/kipi-plugins-docs
    ignore-modules extragear/graphics/kipi-plugins
end module-set
```

2.7 Avslutning av komma igång

Detta är huvudfunktionerna och koncepten som behövs för att komma igång med kdesrc-build

För ytterligare information kan man fortsätta läsa igenom den här dokumentationen. I synnerhet är listan över kommandoradsväljare och tabellen över inställningsalternativ användbara referenser.

KDE-gemenskapen upprätthåller också en Wiki-referens på nätet om hur man bygger källkoden, som hänvisar till kdesrc-build och inkluderar tips och andra riktlinjer om hur verktyget används.

Kapitel 3

Skriptets funktioner

3.1 Översikt över funktioner

Funktionerna i kdesrc-build omfattar:

- Du kan 'låtsas' att utföra åtgärden. Om du skickar med `--pretend` eller `-p` på kommandoraden, ger skriptet en utförlig beskrivning av de kommandon som ska göras, utan att i själva verket utföra dem. Om du dock aldrig tidigare har kört kdesrc-build, bör du köra kommandot **kdesrc-build `--metadata-only`** först för att `--pretend` ska fungera.

  > TIPS
  > För en ännu utförligare beskrivning av vad kdesrc-build gör, prova att använda väljaren `--debug`.

- kdesrc-build kan (med hjälp av KDE:s FTP-server) tillåta snabb utcheckning av moduler. Om modulen du checkar ut redan har paketerats på hemsidan, laddar kdesrc-build ner den versionen och förbereder den för användning på datorn.

  > TIPS
  > Det finns i regel inget behov för någon särskild förberedelse för att utföra en inledande utcheckning av en Git-modul, eftersom hela Git-arkivet ändå måste laddas ner, så det är lätt för servern att avgöra vad som ska sändas.

  Det är snabbare för dig, och hjälper till att minska lasten på de anonyma Git-servrarna.

- En annan uppsnabbning tillhandahålls genom att starta byggprocessen för en modul så fort källkoden för den modulen har laddats ner (tillgänglig sedan version 1.6).

- Utmärkt stöd för att bygga Qt™-biblioteket (i det fall då KDE-programvaran som du försöker bygga beror på en nyare version av Qt™, som inte är tillgänglig i distributionen).

- kdesrc-build kräver inte att ett grafiskt gränssnitt är tillgängligt för att fungera. Du kan alltså bygga KDE:s programvara utan att behöva en alternativ grafisk miljö.

- Stöder inställning av standardalternativ för alla moduler (som kompileringsinställningarna eller konfigureringsalternativen). Sådana alternativ kan dessutom normalt ändras för specifika moduler.
  Dessutom lägger kdesrc-build till standardflaggor där det är lämpligt, för att spara dig besvär och möjliga fel från att skriva in dem själv. Observera: det gäller dock inte när en (egen) verktygskedja är inställd, t.ex. cmake-toolchain.

- kdesrc-build kan checka ut en särskilt gren eller tagg av en modul. Du kan också försäkra dig om att en specifik revision av en modul checkas ut.
- kdesrc-build kan automatiskt ändra källkodskatalog för att checka ut från ett annat arkiv, gren eller tagg. Det sker automatiskt när du ändrar ett alternativ som ändrar vad arkivets webbadress är, men du måste använda väljaren --src-only för att tala om för kdesrc-build att det är acceptabelt att utföra bytet.
- kdesrc-build kan checka ut delar av en modul, för de situationer där du bara behöver ett program från en stor modul.
- För utvecklare: kdesrc-build kommer att påminna dig om du använder git+ssh:// men ssh-agent inte kör, eftersom det leder till upprepad begäran om lösenord från SSH.
- Kan ta bort byggkatalogen för en modul efter den har installerats för att spara utrymme till bekostnad av framtida kompileringstid.
- Platserna för katalogerna som används av kdesrc-build kan ställas in (till och med per modul).
- Kan använda Sudo, eller ett annat användarspecificerat kommando för att installera moduler så att kdesrc-build inte behöver köras som systemadministratör.
- kdesrc-build kör normalt med reducerad prioritet för att låta dig fortsätta använda datorn medan kdesrc-build arbetar.
- Har stöd för att använda KDE:s taggar och grenar.
- Det finns stöd för att återuppta en byggprocess från en given modul. Du kan till och med ignorera vissa moduler tillfälligt för en given byggprocess.
- kdesrc-build visar förloppet för byggprocessennär CMake används, och tar alltid tid på byggprocessen så att du i efterhand vet hur lång tid den tog.
- Levereras med en inbyggt rimlig uppsättning förvalda alternativ lämpligt för att bygga en grundläggande enanvändarinstallation av KDE från de anonyma källkodsarkiven.
- Expanderar dina inställningsalternativ med tilde. Du kan till exempel ange:

```
qtdir ~/kdesrc/build/qt
```

- Ställer automatiskt in ett byggsystem med källkodskatalogen skild från byggkatalogen, för att hålla källkodskatalogen orörd.
- Du kan ange allmänna alternativ att använda för alla moduler som checkas ut, och du kan dessutom ange alternativ som ska användas för enskilda moduler.
- Tvinga fullständig omkompilering genom att köra kdesrc-build med väljaren `--refresh-build`.
- Du kan ange diverse miljövariabler som ska användas under byggprocessen, inklusive `KDEDIR`, `QTDIR`, `DO_NOT_COMPILE` och `CXXFLAGS`.
- Kommandologgning. Loggar dateras och numreras så att du alltid har en logg av en körning av skriptet. Dessutom skapas en symbolisk länk som heter latest, som alltid pekar på den senaste loggen i loggkatalogen.

3.2 Byggloggning i kdesrc-build

3.2.1 Översikt över loggning

Loggning är en funktion i kdesrc-build varigenom utmatningen från varje kommando som kdesrc-build kör sparas i en fil för senare undersökning, om nödvändigt. Det görs eftersom det ofta är nödvändigt att ha utmatningen från programmen när ett byggfel uppstår, eftersom det finns så många orsaker att en byggprocess kan misslyckas från början.

3.2.1.1 Loggningskatalogens layout

Loggarna lagras alltid i loggkatalogen. Platsen för loggkatalogen styrs av alternativet log-dir, som normalt är `${source-dir} /log` (där ${source-dir} är värdet av alternativet source-dir. I resten av avsnittet kallas värdet ${log-dir}).

I ${log-dir} finns en uppsättning kataloger, en för varje gång kdesrc-build kördes. Varje katalog namnges med datum och körnumret. Exempelvis, den andra gången kdesrc-build körs 26:e maj, 2004, skulle en katalog som heter `2004-05-26-02` skapas, där 2004-05-26 står för datumet och -02 är körnumret.

För din bekvämlighet skapar kdesrc-build också en länk till loggarna för den senaste körningen, som heter `latest`. Loggarna för den senaste körningen av kdesrc-build ska alltid finnas under `${log-dir} /latest`.

Varje katalog för en körning av kdesrc-build innehåller nu i sin tur en uppsättning kataloger, en för varje KDE-modul som kdesrc-build försöker bygga. Dessutom finns en fil som heter `build-status` i katalogen, som låter dig avgöra vilka moduler som byggdes och vilka som misslyckades.

> **NOT**
> Om en modul själv har en delmodul (såsom extragear/multimedia, playground/utils eller KDE/kdelibs), finns i själva verket motsvarande struktur i loggkatalogen. Loggarna för KDE/kdelibs efter den senaste körningen av kdesrc-build finns till exempel i `${log-dir} /latest/KDE/kdelibs`, och inte i `${log-dir} /latest/kdelibs`.

Du hittar en uppsättning filer för varje åtgärd som kdesrc-build utför i varje modulloggkatalog. Om kdesrc-build uppdaterar en modul, kan du se filnamn såsom `git-checkout-update.log` (för en utcheckning av en modul eller när en modul uppdateras som redan har checkats ut). Om kommandot **configure** kördes, kan du förvänta dig att se filen `configure.log` i katalogen.

Om ett fel uppstod, bör du kunna se en förklaring av varför i en av filerna. För att hjälpa till att avgöra vilken fil som innehåller felet, skapar kdesrc-build en länk från filen med felet (såsom `build-1.log` till en fil som heter `error.log`).

Slutresultatet av allt detta är att för att se varför en modul misslyckades byggas efter den senaste körningen av kdesrc-build, är filen du först ska titta i `${log-dir} /latest/ modulnamn /error.log`.

> **TIPS**
> Om filen `error.log` är tom (särskilt efter en installation), kanske det inte var något fel. Vissa verktyg som används av KDE:s byggsystem rapporterar ibland av misstag ett fel när det inte fanns något.
> Dessutom kringgår några kommandon omdirigeringen av utmatning som kdesrc-build gör, och går förbi loggfilen under vissa omständigheter (normalt när den första utcheckningen från Git sker), och felutmatningen finns inte i loggfilen i detta fall, utan istället i terminalen där kdesrc-build kördes

Kapitel 4

Anpassa kdesrc-build

4.1 Översikt av kdesrc-build anpassning

För att använda skriptet måste du ha en fil i din hemkatalog som heter `.kdesrc-buildrc`, som beskriver modulerna som du vill ladda ner och bygga, och eventuella alternativ eller inställningsparametrar att använda för dessa moduler.

4.1.1 Inställningsfilens layout

4.1.1.1 Allmän inställning

Inställningsfilen börjar med allmänna alternativ, angivna enligt följande:

```
global
alternativ-namn alternativ-värde
[...]
end global
```

4.1.1.2 Modulinställning

Den följs därefter av en eller flera modulsektioner, angivna på ett av följande två sätt:

- ```
  module modul-namn
  alternativ-namn alternativ-värde
  [...]
  end module
  ```
- ```
  module-set module-set-name
    repository kde-projects eller git://host.org/path/to/repo.git
    use-modules modulnamn

  # Andra alternativ kan också anges
  alternativnamn alternativvärde
  [...]
  end module-set
  ```

> **VIKTIGT**
> Observera att det andra sättet, med moduluppsättningar, *bara fungerar för Git-baserade moduler*.

För Git-moduler måste *`modul-namn`* vara en modul i KDE:s Git-arkiv (till exempel kdeartwork eller kde-wallpapers), även om det går att undvika det om du anger webbadressen i Git manuellt.

För Git-moduler kan modulnamnet vara i stort sett vad du vill, under förutsättning att det inte är en dubblett av något annat modulnamn i inställningen. Kom ihåg att källkod- och byggkatalogens layout kommer att baseras på modulnamnet om du inte använder väljaren dest-dir.

För Git *moduluppsättningar* måste dock *`modulnamn`* motsvara verkliga Git-moduler i det valda arkivet. Se git-repository-base eller use-modules för mer information.

4.1.1.3 Behandling av alternativvärden

I allmänhet används hela radens innehåll efter *`alternativnamn`* som *`alternativvärde`*.

En modifikation som kdesrc-build utför är att följden **`${alternativnamn}`** ersätts med värdet av det alternativet från den globala inställningen. Det låter dig referera till värden på befintliga alternativ, inklusive alternativ som redan ställts in av kdesrc-build.

För att se ett exempel på detta i användning, se Exempel 2.1.

4.1.1.4 'options'-moduler

Det finns en sista typ av post i inställningsfilen, `options`-grupper, som kan anges var som helst där `module` eller `module-set` kan användas.

```
options modul-namn
alternativ-namn alternativ-värde
[...]
end options
```

En `options`-grupp kan ha inställda alternativ precis som en moduldeklaration, och är kopplad till en befintlig modul. Alla alternativ som ställs in på detta sätt används för att *överskrida* alternativ inställda för den tillhörande modulen.

> **VIKTIGT**
> Den tillhörande modulens namn *måste* stämma med namnet som anges i deklarationen av `options`. Var försiktig så att inte ett felstavat namn skrivs in.

Detta är användbart för att göra det möjligt att deklarera en hel `module-set` med moduler, som alla använder samma alternativ, och därefter använda `options`-grupper för att göra individuella ändringar.

Gruppen `options` kan också gälla namngivna moduluppsättningar. Det låter expertanvändare använda en gemensam inställningsfil (som inkluderar deklarationer av `module-set`) som grundinställning, och sedan göra ändringar av alternativen som används av sådana moduluppsättningar i inställningsfiler med kommandot `include` för att referera till grundinställningen.

Example 4.1 Exempel på användning av alternativ

I det här exemplet väljer vi att bygga alla moduler från KDE:s multimediaprogramvara. Dock vill vi använda en annan version av programmet KMix (kanske för att prova en felrättning). Det fungerar på följande sätt:

```
module-set kde-multimedia-set
  repository kde-projects
  use-modules kde/kdemultimedia
  branch master
end module-set

# kmix är en del av kde/kdemultimedia group, även om vi aldrig namngav
# kmix tidigare i den här filen, räknar kdesrc-build ut ändringen.
options kmix
  branch KDE/4.12
end options
```

När kdesrc-build nu körs, byggs alla KDE:s multimediaprogram från grenen 'master' i källkodsarkivet, men KMix byggs från den äldre grenen 'KDE/4.12'. Genom att använda `options`, behövde man inte lista alla de *andra* programmen i KDE:s multimedia individuellt för att ge dem rätt grenalternativ.

NOT
Observera att funktionen är bara tillgänglig i kdesrc-build från version 1.16, eller genom att använda utvecklingsversionen av kdesrc-build efter 2014-01-12.

4.1.2 Inkludera andra inställningsfiler

Inne i inställningsfilen kan du referera till andra filer genom att använda nyckelordet `include` med en fil, vilket fungerar som om den refererade filen hade infogats i inställningsfilen på det stället.

Du skulle exempelvis kunna göra något liknande:

```
global
    include ~/common-kdesrc-build-options

    # Infoga specifika alternativ här.

end global
```

NOT
Om du inte anger fullständig sökväg för filen som ska inkluderas, eftersöks filen med början i katalogen som innehåller källkodsfilen. Det fungerar också rekursivt.

4.1.3 Ofta använda inställningsalternativ

Följande är en lista med alternativ som ofta används. Klicka på alternativet för att ta reda på mer om det. För att visa den fullständiga listan med alternativ, se Avsnitt 4.2.

- cmake-options för att ange vilka flaggor en modul ska ställas in med vid användning av CMake.

- branch, för att checka ut från en gren istället för `master`.
- configure-flags för att ange vilka flaggor Qt™ ska ställas in med.
- kdedir, för att ange katalogen där KDE ska installeras.
- make-options, för att skicka väljare till byggprogrammet Make (såsom antal processorer att använda).
- qtdir, för att ange sökvägen till Qt™.
- source-dir, för att ändra vart källkoden laddas ner.

4.2 Tabell av tillgängliga inställningsalternativ

Här är en tabell med diverse alternativ, som innehåller följande information:

- Alternativets namn
- En beskrivning av hur kdesrc-build beter sig om alternativet både är angivet i den allmänna sektionen och modulsektionen i inställningsfilen när modulen byggs.
- Särskilda kommentarer om alternativets syfte och användning.

Alternativnamn	**Modul -> Allmänt beteende**	**Anmärkningar**
async	Kan inte överskridas	Alternativet aktiverar asynkront användningssätt, där uppdatering av källkoden och byggprocessen utförs i parallell, istället för att vänta på alla uppdateringar av källkoden innan byggprocessen startas. Alternativets förvalda värde är att aktivera asynkront användningssätt. För att inaktivera, ställ in alternativet till **`false`**. Alternativet är tillgängligt från och med utgåva 1.6.

binpath	Modulinställning överskrider allmän	Ange alternativet för att sätta miljövariabeln PATH under byggprocessen. Du kan inte överskrida alternativet i ett modulalternativ. Förvalt värde är `$PATH` som är inställd när skriptet startar. Miljövariabeln ska innehålla sökvägar till utvecklingsverktygen åtskilda med kolon. Sökvägarna `$KDEDIR /bin` och `$QTDIR /bin` läggs automatiskt till. Du kan använda tilde (`~`) för eventuella sökvägar du lägger till med det här alternativet.
branch	Modulinställning överskrider allmän	Ställ in alternativet för att checka ut från en gren av KDE istället för standardvärdet *`master`* där utveckling av KDE sker. För att till exempel checka ut grenen KDE 4.6, skulle du ställa in alternativet till *`4.6`*. Om kdesrc-build misslyckas med att ladda ner en gren med väljaren, kan du behöva ange webbadressen att ladda ner från för hand med alternativet module-base-path eller override-url. NOT För de flesta KDE-moduler vill man troligtvis använda alternativet branch-group istället och använda det här alternativet för undantag från fall till fall.

branch-group	Modulinställning överskrider allmän	Ställ in alternativet till en allmän grupp som du vill att moduler ska väljas från. För modultyper i Git som stöds, bestämmer kdesrc-build den verkliga grenen att använda baserat på regler kodade av KDE-utvecklarna (reglerna kan ses i källkodsarkivet `kde-build-metadata` i din källkodskatalog). Efter att en gren har bestämts används den som om du hade angivit den själv med alternativet branch. Det är användbart om du bara försöker upprätthålla senaste version för ett antal normala utvecklingsspår utan att behöva ta hänsyn till alla ändringar av namn på grenar. Observera att om du *själv* använder `branch` överskrider det den här inställningen. Samma sak gäller andra specifika alternativ för att välja gren såsom tag. Väljaren lades till i kdesrc-build 1.16-pre2. NOT Väljaren gäller bara Git moduler i `kde-projects` (det vanliga fallet). Se också Avsnitt 2.6.4.

build-dir	Modulinställning överskrider allmän	Använd det här alternativet för att byta katalog som innehåller källkoden att bygga. Det finns tre olika sätt att använda det: 1. Relativt till KDE:s Git-källkodskatalog (se alternativet source-dir). Det är förvalt, och väljes om du skriver in ett katalognamn som inte börjar med tilde (~) eller snedstreck (/). Förvalt värde är `build` 2. Absolut sökväg. Om du anger en sökväg som börjar med /, används sökvägen direkt. Till exempel `/tmp/kde-obj-dir/` 3. Relativt till din hemkatalog. Om du anger en sökväg som börjar med ~, används sökvägen relativt till din hemkatalog, på motsvarande sätt som skalets expansion av tilde. Till exempel skulle `~/builddir` ställa in byggkatalogen till `/home/användarnamn/builddir` Alternativet kan, kanske något oväntat, ändras per modul.

build-when-unchanged	Modulinställning överskrider allmän	Använd alternativet för att bestämma om kdesrc-build alltid försöker bygga en modul som inte har några uppdateringar av källkoden. Genom att ställa in `build-when-unchanged` till **`true`**, försöker alltid kdesrc-build utföra byggfasen för en modul, även om modulen inte har några uppdateringar av källkoden. Det är standardinställningen, eftersom det är troligare att den leder till en korrekt byggprocess. Genom att ställa in `build-when-unchanged` till **`false`**, försöker kdesrc-build bara utföra byggfasen för en modul, om modulen har en uppdateringar av källkoden, eller i andra situationer när det är troligt att det krävs. Det kan spara tid, särskilt om kdesrc-build körs dagligen, eller ännu oftare. VIKTIGT Funktionen tillhandahålls enbart som en optimering. Liksom många andra optimeringar-, finns det överväganden gällande installationens riktighet. Exempelvis kan ändringar av modulen qt eller kdelibs leda till att andra moduler måste byggas om, även om källkoden inte ändras alls.

cmake-generator	Modulinställning överskrider allmän	Använd alternativet för att ange vilken generator som ska användas med CMake. För närvarande stöds både `Ninja` och `Unix Makefiles` samt extra generatorer baserade på dem såsom `Eclipse CDT4 - Ninja`. Ogiltiga värden (som inte stöds) ignoreras, och behandlas som inte angivet. Om inte angivet används `Unix Makefiles` som förval. Observera att om en giltig generator också anges via cmake-options överskrider den värdet på `cmake-generator`.
cmake-toolchain	Modulinställning överskrider allmän	Använd alternativet för att ange en verktygskedjefil att använda med CMake. När en giltig verktygskedjefil är inställd, kommer kdesrc-build *inte ställa in miljövariabler automatiskt*. Du kan använda set-env, binpath och libpath för att manuellt rätta miljön om verktygskedjefilen inte fungerar som den är med kdesrc-build. Se översikten över standardflaggor som läggs till av kdesrc-build för mer information. Observera att om en giltig verktygskedja också anges via cmake-options överskrider den värdet på `cmake-toolchain`.

cmake-options	Lägger till globala alternativ för det normala byggsystemet, överskrider global för andra byggsystem.	Använd det här alternativet för att ange vilka flaggor som ska skickas till CMake när byggsystemet för modulen skapas. När det används som ett allmänt alternativ, används det för alla moduler som skriptet bygger. När det används som ett modulalternativ, läggs det till i slutet av de allmänna alternativen. Det låter dig ange gemensamma CMake-alternativ i den allmänna delen. Alternativet gäller inte för qt (som inte använder CMake). Använd configure-flags istället. Om en giltig generator anges bland de listade alternativen överskrider den värdet på cmake-generator. Ogiltiga generatorer (som inte stöds) ignoreras och skickas inte till CMake. Om en giltig verktygskedja anges bland de listade alternativen överskrider den värdet på cmake-toolchain. Ogiltiga verktygskedjor ignoreras och skickas inte till CMake. Eftersom alternativen skickas direkt till kommandoraden för CMake, ska de anges som de skulle skrivits till CMake. Till exempel: `cmake-options -DCMAKE_BUILD_TYPE=RelWithDebInfo` Eftersom det är krångligt, anstränger sig kdesrc-build för att så länge övriga alternativ är rätt inställda, ska du kunna lämna det här alternativet tomt. (Med andra ord, *nödvändiga* parametrar för CMake ställs in automatiskt åt dig.)

colorful-output	Kan inte överskridas	Ställ in alternativet till **`false`** för att inaktivera den färgglada utmatningen från kdesrc-build. Alternativet har standardvärdet `true`. Observera att kdesrc-build inte matar ut färgkoder till något annat än en terminal (såsom xterm, Konsole eller den vanliga konsollen i Linux®).
compile-commands-export	Modulinställning överskrider allmän	Aktiverar generering av `compile_commands.json` via CMake i byggkatalogen. Alternativet har förvalt värde `true`, ställ in det till **`false`** för att inaktivera beteendet.
compile-commands-linking	Modulinställning överskrider allmän	Aktiverar att skapa symboliska länkar från `compile_commands.json` genererad via CMake i byggkatalogen till motsvarande källkodskatalog. Alternativet har förvalt värde `false`, ställ in det till **`true`** för att aktivera att den symboliska länken automatiskt skapas.
configure-flags	Lägger till globala alternativ för det normala byggsystemet, överskrider global för andra byggsystem.	Använd det här alternativet för att ange vilka flaggor som ska skickas till ./configure när byggsystemet för modulen skapas. När det används som ett allmänt alternativ, används det för alla moduler som skriptet bygger. *Alternativet fungerar bara för qt.* För att ändra konfigurationsinställningar för KDE-moduler, se cmake-options.

custom-build-command	Modulinställning överskrider global (alternativ i byggsystemet)	Alternativet kan anges för att köra ett annorlunda kommando (exempelvis annat än **make**) för att utföra byggprocessen. kdesrc-build bör i allmänhet göra rätt, så alternativet ska inte behövas anges. Det kan dock vara användbart för att använda alternativa byggsystem. Alternativets värde används som kommandoraden att köra, ändrad av make-options som vanligt.
cxxflags	Lägger till globala alternativ för det normala byggsystemet, överskrider global för andra byggsystem.	Använd det här alternativet för att ange vilka flaggor som ska användas för att bygga modulen. Alternativet anges här istället för med configure-flags eller cmake-options eftersom alternativet också sätter miljövariabeln `CXXFLAGS` under byggprocessen. Observera att för KDE 4 och alla andra moduler som använder CMake, är det nödvändigt att ställa in alternativet CMAKE_BUILD_TYPE option till **`none`** när modulen konfigureras. Det kan göras med alternativet cmake-options.
dest-dir	Modulinställning överskrider allmän	Använd alternativet för att ändra namnet en modul får på disk. Om modulen till exempel är extragear/network, skulle du kunna ändra det till extragear-network med det här alternativet. Observera att även om detta ändrar modulens namn på disk, är det inte en god idé att låta kataloger eller katalogåtskiljare ingå i namnet, eftersom det kommer i konflikt med eventuella build-dir eller source-dir alternativ.

disable-agent-check	Kan inte överskridas	Om du använder SSH för att ladda ner källkod från Git (om du använder protokollet git+ssh), försöker kdesrc-build normalt att försäkra sig om att när du använder ssh-agent, så hanterar det verkligen några SSH-identiteter. Det görs för att försöka förhindra att SSH frågar efter din lösenordsfras för varje modul. Du kan inaktivera kontrollen genom att sätta `disable-agent-check` till **`true`**.
do-not-compile	Modulinställning överskrider allmän	Använd alternativet för att välja en specifik uppsättning kataloger att inte bygga i en modul (istället för alla). Katalogerna att inte bygga ska skiljas åt med mellanslag. Observera att källkoden till programmen fortfarande laddas ner. För att exempelvis inaktivera bygge av katalogerna `codeeditor` och `minimaltest` i ramverket `syntaxhighlighting`, skulle du lägga till **`do-not-compile codeeditor minimaltest`**, och lägga till ″do-not-compile juk kscd″ i alternativen för syntaxfärgläggning. Se Avsnitt 6.3.1.1 för ett exempel.

git-desired-protocol	Kan inte överskridas	Väljaren gäller bara moduler från ett arkiv som ingår i KDE-projektet. Vad väljaren i själva verket gör är att ställa in vilket nätverksprotokoll som ska föredras när källkod för dessa moduler laddas upp. Normalt används det mycket effektiva protokollet `git`, men det kan vara blockerat i vissa nätverk (t.ex. interna företagsnätverk, öppna wifi-nätverk). Ett alternativt protokoll som stöds mycket bättre är protokollet `https` som används för webbplatser på Internet. Om du använder ett av dessa begränsade nätverk kan du ändra väljaren till **`http`** för att föredra kommunikation med `https` istället. TIPS Du kan också behöva väljaren http-proxy, om en HTTP-proxy också behövs för nätverkstrafik. I alla andra situationer ska väljaren inte användas, eftersom standardprotokollet är det effektivaste. Väljaren lades till i kdesrc-build 1.16. Innan 20.06 användes väljaren för att ställa in hämtningswebbadressen istället för uppladdningswebbadressen. Från 20.06 används alltid `https` när KDE-projekt uppdateras.

git-repository-base	Kan inte överskridas	Det här alternativet, tillagt i version 1.12.1, används för att skapa ett kort namn för att hänvisa till en specifik baswebbadress för ett Git-arkiv i senare deklarationer av moduluppsättningar, vilket är användbart för att snabbt deklarera många Git-moduler att bygga. Två saker måste anges (åtskilda med ett mellanslag): Namnet som baswebbadressen ska tilldelas till, och själva baswebbadressen. Till exempel:
		
		`global`
		`  # Andra alternativ`
		
		`  # Det här är den gemensamma sökvägen till al`
		`  git-repository-base kde-gitkde:`
		`end global`
		
		`# Moduldeklarationer`
		
		`module-set`
		`  # Nu kan det alias som definierades tidigare`
		`  # barai en moduluppsättning.`
		`  repository kde-git`
		
		`  use-modules module1.git module2.git`
		`end module-set`
		
		Moduluppsättningarnas alternativ `use-modules` skapar två moduler internt, och kdesrc-build beter sig som om det hade läst in:
		
		`module module1`
		`  repository kde:module1.git`
		`end module`
		
		`module module2`
		`  repository kde:module2.git`
		`end module`
		
		Prefixet `kde:` för Git-arkiv som används ovan är en genväg som automatiskt ställs in av kdesrc-build. Se teknikbasens artikel URL Renaming för mer information. Observera att i motsats till de flesta andra

git-user	Modulinställning överskrider allmän	Alternativet är avsett för KDE-utvecklare. Om det anges, används det för att automatiskt ställa in identitetsinformation för källkodshanteringssystemet Git för *nyligen nerladdade* Git-moduler (inklusive det stora flertalet KDE-moduler). Specifikt fylls användarens namn och e-postfält i till de värden som anges med alternativet för varje nytt Git-arkiv. Värdet måste anges på formen `Användarnamn <e-post@exempel.se>`. Exempelvis skulle en utvecklare vid namn 'Anna Johansson' med e-postadressen 'anna@exempel.se' använda: `git-user Anna Johansson <anna@exempel.se>` Alternativet introducerades i kdesrc-build 15.09.
http-proxy	Modulinställning överskrider allmän	Alternativet, om det anges, använder den givna webbadressen som proxyserver för all nätverkskommunikation via HTTP (till exempel när sparade versioner laddas ner för nya moduler, eller KDE:s projektdatabas). Dessutom försöker kdesrc-build försäkra sig om att verktygen det beror på också använder proxyservern, om möjligt, genom att ställa in miljövariabeln `http_proxy` till den angivna servern, *om denna miljövariabel inte redan finns*. Alternativet lades till i kdesrc-build 1.16.

directory-layout	Modulinställning överskrider allmän	Alternativet används för att ställa in den layout som kdesrc-build ska använda när källkods- och byggkataloger skapas. För närvarande finns tre möjliga värden: **`flat`**, **`invent`** och **`metadata`** Layouten **`flat`** är förvalt värde grupperar alla moduler direkt under källkods- och byggkatalogerna på toppnivå. Exempelvis skulle `source/extragear/network/telepathy/ktp-text-ui` med layouten**`metadata`** vara `source/ktp-text-ui` vid användning av layouten **`flat`** istället. Layouten **`invent`** skapar en kataloghierarki som speglar de relativa sökvägarna för arkiv på invent.kde.org. Exempelvis skulle `source/kde/applications/kate` med layouten **`metadata`** vara `source/utilities/kate` vid användning av layouten **`invent`** istället. Layouten påverkar bara KDE-projekt. Det är ett bra val för folk som börjar använda kdesrc-build. Slutligen är layouten **`metadata`** samma som det tidigare förvalda beteendet. Layouten organiserar KDE-projekt enligt projektsökvägarna specificerade i projektets metadata för modulerna. Det är ett bra val om du vill ha en kataloglayout som följer vissa KDE-processer, men observera att sökvägen därför inte alltid är stabil. Resultatet kan bli att kdesrc-build överger en gammal kopia av arkivet och skapar en dubblett för ett projekt på grund av ändringar av projektets metadata.

ignore-modules	Kan inte överskridas	Moduler som namnges av alternativet, som skulle ha valts av kdesrc-build på grund av alternativet use-modules, hoppas istället över helt. Använd alternativet när du vill bygga en hel gruppering av kde-project projekt *utom* några specifika moduler. Alternativets värde behöver inte nödvändigtvis namnge modulen direkt. Vilken modul som helst där fullständiga delar i följd av dess KDE projektmodul-sökväg matchar ett av alternativets värden ignoreras, så det går alltså att ignorera flera moduler på detta sätt. Exempelvis skulle alternativvärdet `libs` göra att både kde/kdegraphics/libs och playground/libs skulle exkluderas (dock inte kde/kdelibs eftersom den fullständiga delen 'kdelibs' är det som jämförs med). TIPS Se också Exempel 2.7. Alternativet lades till i kdesrc-build 1.16.

include-dependencies	Modulinställning överskrider allmän	När alternativet är **`true`**, begär det att kdesrc-build också ska inkludera den här modulens kända beroenden i bygget, utan att kräva att du nämner dessa beroenden (inte ens indirekt). NOT Alternativet fungerar bara för moduler baserade på `kde-project`, och kräver att metadata som underhålls av KDE-utvecklare är korrekt för vald branch-group. Alternativet är normalt aktiverat för att stödja att bygga program som behöver versioner av Qt™ eller Plasma som är senare en de som stöds av vanliga operativsystem.
install-after-build	Modulinställning överskrider allmän	Alternativet används för att installera paketet efter det har byggts med lyckat resultat. Alternativet är normalt aktiverat. Om du vill inaktivera det, måste du ställa in alternativet till **`false`** i inställningsfilen. Du kan också använda kommandoradsväljaren `--no-install`.

install-environment-driver	Kan inte överskridas	Normalt försöker kdesrc-build installera ett skalskript som kan anropas i användarens profilinställningsskript för att enkelt upprätta nödvändiga miljövariabler för att köra Plasma-skrivbordet som byggts av kdesrc-build. Drivrutinen ändra följande filer: • `$XDG_CONFIG_HOME/kde-env-master.sh` (som normalt finns som `~/.config/kde-env-master.sh`). • `$XDG_CONFIG_HOME/kde-env-user.sh` (som normalt finns som `~/.config/kde-env-user.sh`). Skriptet `kde-env-user.sh` är valfritt. Det är avsett för användaranpassning (se avsnittet Troubleshooting and Debugging på KDE:s användarbas för exempel på anpassningsbara inställningar), men inställningarna kan göras på andra ställen av användaren i befintliga profilinställningsskript. Funktionen kan inaktiveras genom att ställa in alternativet till *`false`*, och försäkra att också alternativet install-session-driver är inaktiverat. Alternativet lades till i kdesrc-build 17.08. **TIPS** kdesrc-build skriver inte över befintliga filer om du inte också anger kommandoradsväljaren `--delete-my-settings`.

install-session-driver	Kan inte överskridas	Om aktiverat, försöker kdesrc-build installera en drivrutin för den grafiska inloggningshanteraren som låter dig logga in på KDE-skrivbordet som byggts med kdesrc-build. Drivrutinen ändra följande filer: • `~/.xsession` • `$XDG_CONFIG_HOME/kde-env-master.sh` (som normalt finns som `~/.config/kde-env-master.sh`). • `$XDG_CONFIG_HOME/kde-env-user.sh` (som normalt finns som `~/.config/kde-env-user.sh`). Om du hanterar en egen inloggningshanterare kan du inaktivera funktionen genom att ställa in alternativet till *`false`*. Om aktiverad, aktiverar funktionen också funktionen install-environment-driver. Alternativet lades till i kdesrc-build 1.16. **TIPS** kdesrc-build skriver inte över befintliga filer om du inte också anger kommandoradsväljaren `--delete-my-settings`.
kdedir	Modulinställning överskrider allmän	Det här alternativet anger katalogen som KDE installeras i efter det har byggts. Förvalt värde är `~/kde`. Om du ändrar det till en katalog som kräver åtkomst som systemadministratör, bör du också läsa om alternativet make-install-prefix.

libname	Modulinställning överskrider allmän	Ställ in det här alternativet för att ändra standardnamnet på det installerade bibliotekskatalogen inuti $KDEDIR och $QTDIR. På många system är detta antingen ″lib″ eller ″lib64″. Automatisk identifiering försöker göras för att ställa in rätt namn som förval, men om gissningen är fel så kan det ändras med inställningen.
libpath	Modulinställning överskrider allmän	Ange alternativet för att sätta miljövariabeln LD_LIBRARY_PATH under byggprocessen. Du kan inte överskrida inställningen med ett modulalternativ. Förvalt värde är tomt, men sökvägarna $KDEDIR /$LIBNAME och $QTDIR /$LIBNAME läggs automatiskt till. Du kan använda tilde (~) i alla sökvägar du lägger till med alternativet.
log-dir	Modulinställning överskrider allmän	Använd det här alternativet för att ändra katalogen som används för att lagra loggfiler som skapas av skriptet.
make-install-prefix	Modulinställning överskrider allmän	Sätt den här variabeln till en lista åtskilda med mellanslag, som tolkas som ett kommando och dess väljare för att föregå kommandot **make install** som används för att installera moduler. Det är användbart för att till exempel installera paket med Sudo, men var försiktig när du hanterar systemadministratörsrättigheter.

make-options	Modulinställning överskrider global (alternativ i byggsystemet)	Ange den här variabeln för att skicka kommandoradsväljare till kommandot **make**. Det är användbart för program som distcc eller system med mer än en processorkärna. Observera att inte alla byggsystem som stöds använder **make**. För byggsystem som använder **ninja** för att bygga (såsom byggsystemet Meson, se inställningen ninja-options.
manual-build	Modulinställning överskrider allmän	Sätt alternativvärdet till **`true`** för att förhindra att byggprocessen försöker bygga modulen. Den hålls fortfarande uppdaterad vid uppdatering från Git. Alternativet motsvarar exakt kommandoradsväljaren `--no-build`.
manual-update	Modulinställning överskrider allmän	Sätt alternativvärdet till **`true`** för att förhindra att byggprocessen försöker uppdatera (och som en följd, bygga eller installera) modulen. Om du anger alternativet för en modul, är det i stort sett samma som att kommentera bort den.

module-base-path	Modulinställning överskrider allmän	Sätt det här alternativet för att överskrida kdesrc-builds förvalda katalogväg till modulen i fråga. Det kan till exempel användas för att hämta specifika grenar eller taggade versioner av bibliotek. KDE:s källkodsvisning är ovärderlig som hjälp för att välja rätt sökväg. Observera att kdesrc-build konstruerar den slutliga sökvägen enligt följande mall: `$repository /home/kde/ $module-base-path`. Förvalt värde är antingen `$module` eller `KDE/ $module`, beroende på modulnamnet. **TIPS** Använd alternativen branch eller tag istället så snart de är tillämpliga.
niceness	Kan inte överskridas	Ställ in alternativet till ett tal mellan 20 och 0. Ju högre nummer, desto lägre prioritet ställer kdesrc-build in åt sig själv, dvs. ju högre numret är desto ”snällare” är programmet. Förvalt värde är 10.

ninja-options	Modulinställning överskrider global (alternativ i byggsystemet)	Ställ in variabeln för att skicka kommandoradsväljare till byggkommandot **ninja**. Det kan vara användbart för att aktivera 'detaljerad' utmatning eller för att manuellt reducera antal byggjobb som **ninja** skulle använda.
		
		NOT
		Observera att inställningen bara styr ninja när det används med kdesrc-build. Qt™-modulen 'webengine' använder **ninja** indirekt, men stöder bara officiellt att byggas med **make**. I detta fall kan `NINJAFLAGS` ställas in som ett sätt att låta **make** skicka lämpliga flaggor när det senare anropar **ninja**, genom att använda make-options.
		
		`options qtwebengine`
		`  # Begränsa make och ninja att inte använda`
		`  # även om fler processorer är tillgängliga`
		`  make-options -j6 NINJAFLAGS=-j6`
		`end options`

no-src	Modulinställning överskrider allmän	Om alternativet är sant, uppdaterar inte kdesrc-build modulens källkod automatiskt. Det försöker i alla fall bygga modulen om det normalt ändå hade försökt göra det.
num-cores	Kan inte överskridas	Alternativet definieras av kdesrc-build (när verktyget kdesrc-build-setup används eller **kdesrc-build --initial-setup**) till antal tillgängliga processorer (som indikeras av det externa programmet nproc). Om kdesrc-build inte kan detektera antal processorer, ställs värdet in till 4. Se Exempel 2.1 för ett exempel på användning av alternativet. Alternativet lades till i version 20.07.

num-cores-low-mem	Kan inte överskridas	Alternativet definieras av kdesrc-build (när verktyget kdesrc-build-setup används eller **kdesrc-build --initial-setup**) till antal processorer som anses vara säkert för tunga eller andra högintensiva moduler, såsom `qtwebengine`, för att undvika att minnet tar slut under bygget. Den typiska beräkningen är en processorkärna per 2 Gibibyte (GiB) totalt minne. Åtminstone en kärna anges, och inte fler än `num-cores`. Även om alternativet är avsett att stödja Qt™-moduler, kan det användas för vilken modul som helst på samma sätt som `num-cores` används. Om kdesrc-build inte kan detektera tillgängligt minne ställs värdet in till 2. Alternativet lades till i version 20.07.

override-build-system	Modulinställning överskrider allmän	Det här är en avancerad väljare, tillagd i kdesrc-build 1.16. Normalt detekterar kdesrc-build lämpligt byggsystem att använda för en modul när den har laddats ner. Det görs genom att kontrollera att specifika filer finns i modulens källkodskatalog. Vissa moduler kan innehålla mer än en uppsättning filer som krävs, vilket skulle kunna förvirra den automatiska detekteringen. I detta fall kan du ange rätt byggtyp manuellt. De byggtyper som för närvarande stöds och kan anges är: **KDE** Används för att bygga KDE-moduler. I själva verket kan den användas för att bygga nästan vilken modul som helst som använder CMake, men det är bäst att inte förlita sig på det. **Qt** Används för att bygga själva Qt™-biblioteket. **qmake** Används för att bygga Qt™-moduler som använder `.pro`-filer i enlighet med qmake. **generic** Används för att bygga moduler som använder enkla Makefiles och inte kräver någon särskild konfigurering.

override-url	Modulinställning överskrider allmän	Om du anger det här alternativet, använder kdesrc-build dess värde som webbadress att skicka till Git*helt oförändrad*. Du bör i allmänhet använda det om du vill ladda ner en specifik utgåva, men kdesrc-build inte kan räkna ut vad du menar genom att använda branch.
persistent-data-file	Kan inte överskridas	Använd väljaren för att ändra var kdesrc-build lagrar bestående data. Normalt lagras sådan data i en fil som heter `.kdesrc-build-data` i samma katalog som inställningsfilen som används. Om den allmänna inställningsfilen används, lagras den i `~/.local/state/kdesrc-build-data` (`$XDG_STATE_HOME/kdesrc-build-data`, om `$XDG_STATE_HOME` är angivet). Om du har flera tillgängliga inställningar i samma katalog, kan det vara bra att ställa in alternativet så att de olika inställningarna inte råkar ut för konflikter i bestående data. Väljaren lades till i kdesrc-build 1.15.
prefix	Modulinställning överskrider allmän	Alternativet bestämmer var en modul installeras (normalt används inställningen `kdedir`). Genom att använda alternativet kan du installera en modul i en annan katalog än där KDE:s plattformsbibliotek installeras, om du exempelvis bara använder kdesrc-build för att bygga program. Du kan använda `${MODULE}` eller `$MODULE` i sökvägen för att expandera dem till modulens namn.

purge-old-logs	Modulinställning överskrider allmän	Alternativet bestämmer om gamla loggkataloger automatiskt tas bort eller inte. Standardvärdet är *true*.
qmake-options	Modulinställning överskrider allmän	Alla väljare som anges här skickas till kommandot **qmake**, för moduler som använder byggsystemet qmake. Det går exempelvis att använda väljaren **PREFIX=/sökväg/till/qt** i qmake för att överskrida var modulen installeras. Väljaren lades till i kdesrc-build 1.16.
qtdir	Modulinställning överskrider allmän	Ange alternativet för att sätta miljövariabeln QTDIR under byggprocessen. Om alternativet inte anges, antar kdesrc-build att Qt™ tillhandahålls av operativsystemet.

remove-after-install	Modulinställning överskrider allmän	Om du har ont om hårddiskutrymme kan du vilja använda det här alternativet för att automatiskt ta bort byggkatalogen (eller både källkatalogen och byggkatalogen för engångsinstallationer) efter modulen har installerats med lyckat resultat. Möjliga värden för alternativet är: • none - Ta inte bort någonting (det är normalvärdet). • builddir - Ta bort byggkatalogen men inte källkoden. • all - Ta både bort källkoden och byggkatalogen. Observera att användning av alternativet kan ha en markant effekt både på användning av bandbredd (om du använder `all`) och tiden det tar att kompilera KDE:s programvara, eftersom kdesrc-build inte kommer att kunna utföra inkrementella byggprocesser.
repository	Modulinställning överskrider allmän	Alternativet introducerades i version 1.10, och används för att ange Git-arkivet för att ladda ner modulens källkod. Qt™ (och därför qt) behöver alternativet, samt diverse KDE-moduler som håller på att konverteras att använda Git.

revision	Modulinställning överskrider allmän	Om alternativet ställs in till ett annat värde än 0 (noll), tvingar kdesrc-build uppdateringen att ge modulen exakt den version som anges, även om alternativ som branch gäller. Om modulen redan har angiven version, kommer den inte att uppdateras ytterligare om inte alternativet ändras eller tas bort från konfigurationen. NOT Väljaren fungerade inte för git-baserade moduler (inklusive moduler från kde-projects) innan kdesrc-build version 1.16.
run-tests	Modulinställning överskrider global (alternativ i byggsystemet)	Om satt till **`true`**, byggs modulen med stöd för att köra sin testsvit, och testsviten körs som en del av byggprocessen. kdesrc-build visar en enkel rapport med testresultatet. Det är användbar för utvecklare eller de som vill försäkra sig om att systemet är riktigt inställt.

set-env	Modulinställning överskrider allmän	Det här alternativet accepterar en uppsättning värden åtskilda av mellanslag, där det första värdet är miljövariabeln att sätta, och följande värden är vad du vill sätta variabeln till. För att till exempel sätta variabeln `RONALD` till McDonald, skulle du skriva följande kommando under lämplig sektion: `set-env RONALD McDonald` Alternativet är särskilt på det sätt att det kan upprepas utan att överskrida tidigare inställningar av set-env i samma del av inställningsfilen. På så sätt kan du ställa in mer än en miljövariabel per modul (eller allmänt).
source-dir	Modulinställning överskrider allmän	Det här alternativet används för att ange katalogen på datorn där KDE:s Git-källkod ska lagras. Om du inte anger värdet, är förvalt värde `~/kde/src`. Tilde (`~`) kan utnyttjas för att representera hemkatalogen om alternativet används.
ssh-identity-file	Kan inte överskridas	Sätt alternativet för att bestämma vilken privat SSH nyckelfil används av kommandot **ssh-add** när kdesrc-build laddar ner källkod från arkiv som kräver behörighetskontroll. Se också: Avsnitt 6.4.1. Alternativet lades till i version 1.14.2.
stop-on-failure	Modulinställning överskrider allmän	Ställ in det här alternativvärdet till **`false`** för att låta skriptet fortsätta köra efter ett fel uppstår i bygg- eller installationsprocessen. Normalvärdet är `true`.

tag	Modulinställning överskrider allmän	Använd väljaren för att ladda ner en specifik utgåva av en modul. *Observera:* Chansen är mycket stor att du *inte vill* använda alternativet. Utgåvor av KDE är tillgängliga som komprimerade tar-arkiv från KDE:s nerladdningsplats. NOT Väljaren har bara varit stödd för git-baserade moduler sedan kdesrc-build 1.16.
use-clean-install	Modulinställning överskrider global (alternativ i byggsystemet)	Sätt det här alternativet till **`true`** för att få kdesrc-build att köra **make uninstall** direkt innan **make install** körs. Det kan vara användbart för att försäkra sig om att inte några kvarblivna gamla biblioteksfiler, CMake metadata, etc. kan orsaka problem i långlivade installationer av KDE. Dock fungerar det bara med byggsystem som stöder **make uninstall**. Väljaren lades till i kdesrc-build 1.12, men dokumenterades inte förrän i kdesrc-build 1.16.
use-idle-io-priority	Kan inte överskridas	Det här alternativet, tillagt i kdesrc-build 1.12, gör att en lägre prioritet används för disk och annan användning av in- och utmatning, vilket kan förbättra svarsbenägenheten hos resten av systemet signifikant, till kostnad av något längre körtider för kdesrc-build. Standardvärdet är att det är inaktiverat. För att aktivera lägre diskprioritet, ställ in det till **`true`**.

use-inactive-modules	Kan inte överskridas	När det aktiveras låter det här alternativet kdesrc-build också klona och hämta från arkiv som är markerade som inaktiva. Förval är att det är inaktiverat, för att tillåta inaktiva moduler att ställa in det till **`true`**.

use-modules	Kan bara användas i moduluppsättningar	Det här alternativet, tillagt i kdesrc-build 1.12.1, gör det möjligt att enkelt specificera många olika moduler som ska byggas på samma ställe i inställningsfilen. Det här alternativet *måste* användas med en `moduluppsättning`. Varje identifierare som används i alternativet konverteras internt till en kdesrc-build-modul, med alternativet `repository` inställt till moduluppsättningens arkiv, kombinerat med identifierarens namn för att slutgiltigt kunna bestämma arkivet att ladda ner från. Alla andra alternativ som finns i moduluppsättningen kopieras också till de skapade modulerna utan ändring. Ordningen som modulerna definieras i alternativet är viktigt eftersom det också är ordningen som kdesrc-build kommer att behandla de skapade modulerna vid uppdatering, byggning och installation. Alla moduler definierade i en given moduluppsättning kommer att hanteras innan kdesrc-build går vidare till nästa modul efter moduluppsättningen. Om alternativ för en skapad modul måste ändras, deklarera helt enkelt modulen igen efter den har definierats i moduluppsättningen och ställ in alternativen efter behov. Även om alternativen som modulen använder kommer att ändras på detta sätt, uppdateras och byggs modulen ändå i ordningen som anges av moduluppsättningen (dvs. du kan inte ordna om byggföljden genom att göra det).

Tabell 4.1: Alternativtabell

Kapitel 5

Kommandoradsväljare och miljövariabler

5.1 Användning av kommandoraden

kdesrc-build är konstruerat för att köras på följande sätt:

`kdesrc-build` [`--väljare`...] [*moduler att bygga*...]

Om inga moduler att bygga anges på kommandoraden, bygger kdesrc-build alla moduler som är definierade i inställningsfilen, i den ordning de anges i filen (även om det kan ändras av diverse alternativ i inställningsfilen).

5.1.1 Ofta använda kommandoradsväljare

Den fullständiga listan med kommandoradsväljare anges i Avsnitt 5.3. De oftast använda väljarna omfattar:

`--pretend` (eller `-p`)
: Väljaren gör att kdesrc-build indikerar vilka åtgärder som skulle utföras, utan att verkligen implementera dem. Det kan vara användbart för att försäkra dig om att modulerna som du tror du bygger verkligen kommer att byggas.

`--refresh-build`
: Väljaren tvingar kdesrc-build att bygga de angivna modulerna från en fullständigt ren utgångspunkt. Eventuella befintliga byggkataloger för modulerna tas bort och byggs om. Väljaren är användbar om du får fel när en modul byggs, och krävs ibland när Qt™- eller KDE-bibliotek ändras.

`--no-src`
: Alternativet hoppar över uppdateringsprocessen av källkoden. Du kan vilja använda det om du uppdaterade källkoden nyligen (kanske gjorde du det manuellt, eller körde nyligen kdesrc-build), men ändå vill bygga om vissa moduler.

`--no-build`
: Väljaren liknar `--no-src` ovan, men den här gången hoppas byggprocessen över.

5.1.2 Ange moduler att bygga

I allmänhet är det så enkelt som att ange modulnamnet definierat i inställningsfilen för att ange moduler att bygga. Du kan också ange moduler som ingår i en moduluppsättning, antingen som de namnges under use-modules, eller med hela moduluppsättningen, om du har givit den ett namn.

I det specifika fallet med moduluppsättningar baserade på KDE:s projektdatabas, expanderar kdesrc-build modulnamnets komponenter för att bestämma exakt den modul du vill ha. Exempelvis placerar KDE:s projektposten för kdesrc-build projektet i `extragear/utils/kdesrc-build`. Du måste ange något av följande för att bygga kdesrc-build:

```
% kdesrc-build +extragear/utils/kdesrc-build
% kdesrc-build +utils/kdesrc-build
% kdesrc-build +kdesrc-build
```

> NOT
> Kommandona i föregående exempel inledde modulnamn med +. Det tvingar modulnamnet att tolkas som en modul från KDE:s projektdatabas, även om modulen inte har definierats i inställningsfilen.

Var försiktig med att ange mycket generella projekt (t.ex. `extragear/utils` ensamt), eftersom det kan leda till att ett stort antal moduler byggs. Du bör använda väljaren `--pretend` innan en ny modul byggs för att försäkra dig om att bara de moduler du vill kommer att byggas.

5.2 Miljövariabler som stöds

kdesrc-build använder inte miljövariabler. Om du behöver sätta miljövariabler för bygg- eller installationsprocessen, se alternativet set.-env.

5.3 Kommandoradsväljare som stöds

Skriptet accepterar följande kommandoradsväljare:

--async

Aktiverar asynkront användningssätt som kan utföra uppdateringar av källkoden och bygga moduler samtidigt. Det är förvalt värde, och väljaren behöver bara anges om du har inaktiverat det i inställningarna.

--help **(eller *-h*)**

Visa bara enkel hjälp om skriptet.

--version **(eller *-v*)**

Visa programmets version.

--show-info

Visar information om kdesrc-build och operativsystemet, vilket kan vara användbart i felrapporter eller när hjälp efterfrågas på forum eller e-postlistor.

Tillgängligt sedan version 18.11.

--initial-setup

Låter kdesrc-build utföra en initial engångsinställning nödvändig för att förbereda systemet så att kdesrc-build kan fungera, och så att den nyinstallerade KDE-programvaran kan köra.

Det omfattar:

- Installera kända beroenden (på Linux®-distributioner som stöds)
- Lägga till nödvändiga miljövariabler i `~/.bashrc`
- Ställa in en inställningsfil

Tillgängligt sedan version 18.11.

--author

Visa upphovsmannens kontaktinformation.

--color

Aktivera färglagd utmatning. (Det är förval för interaktiva terminaler.)

--nice=värde

Värdet justerar datorns processorprioritet som begärs av kdesrc-build, och ska vara i intervallet 0-20. 0 är högst prioritet (eftersom det är minst 'snällt'), 20 är lägst prioritet. kdesrc-build har förvalt värde 10.

--no-async

Inaktiverar asynkront användningssätt för uppdateringar. Istället utförs hela uppdateringen innan byggningen startas. Väljaren slöar ner totalprocessen, men om du råkar ut för IPC-fel medan du kör kdesrc-build, prova den här väljaren, och skicka in en felrapport.

--no-color

Inaktivera färglagd utmatning.

--pretend **(eller *-p*)**

kdesrc-build kör igenom uppdaterings- och byggprocessen, men istället för att utföra några åtgärder för att uppdatera eller bygga, skriver ut vad skriptet skulle ha gjort (t.ex. vilka kommandon att köra, steg att utföra, etc.).

> NOT
> Enkla läskommandon (som att läsa information) kan fortfarande utföras, för att göra utmatningen mer relevant (som att simulera om källkod skulle checkas ut eller uppdateras på ett riktigt sätt).

> VIKTIGT
> Alternativet kräver att viss nödvändig metadata är tillgänglig, vilken normalt laddas ner automatiskt, men nerladdningar är inaktiverade i låtsasläge. Om du aldrig har kört kdesrc-build (och därför inte har denna metadata), måste du först köra kommandot **kdesrc-build --metadata-only** för att ladda ner nödvändig metadata.

--quiet **(eller *-q*)**

Var inte så högljudd med utmatningen. Med den här väljaren matas bara det viktigaste ut.

--really-quiet

Mata bara ut varningar och fel.

--verbose

Beskriv i detalj vad som händer, och vad kdesrc-build gör.

--src-only **(eller *-s*)**

Utför bara uppdatering av källkoden.

--build-only

Utför bara byggprocessen.

--install-only

Utför bara installationsprocessen.

--metadata-only

Utför bara processen för nerladdning av metadata. kdesrc-build hanterar det normalt automatiskt, men det går att använda detta för att få kommandoradsväljaren `--pretend` att fungera.

--rebuild-failures

Använd väljaren för att bara bygga moduler som misslyckades byggas vid en tidigare körning av kdesrc-build. Det är användbart om ett väsentligt antal misslyckanden inträffade, blandat med lyckade byggen. Efter att ha rättat problemet som orsakade misslyckade byggen är det enkelt att bara bygga modulerna som tidigare misslyckades.

> NOT
> Observera att listan med 'tidigare misslyckade moduler' nollställs varje gång en körning av kdesrc-build blir klar med några misslyckade moduler. Den nollställs dock inte av ett fullständigt lyckat bygge, så det är möjligt att bygga om några moduler med lyckat resultat och ändå använda väljaren.

Väljaren lades till i kdesrc-build 15.09.

--include-dependencies **(eller *-d*),** ***--no-include-dependencies*** **(eller *-D*)**

Väljaren gör att kdesrc-build automatiskt inkluderar andra KDE- och Qt™-moduler i byggprocessen, om det krävs för modulerna som du har krävt ska byggas på kommandoraden eller i inställningsfilen.

Modulerna som läggs till lagras i KDE:s källkodshanteringssystem. Se Avsnitt 2.6.4.

Motsvarande alternativ i inställningsfilen är include-dependencies.

Du kan också använda `--no-include-dependencies`, som stänger av att automatiskt inkludera ytterligare beroendemoduler.

--ignore-modules **(eller *-!*)**

Inkludera inte moduler som skickas på resten av kommandoraden i uppdaterings- och byggprocessen (det är användbart om du vill bygga de flesta modulerna i inställningsfilen och bara hoppa över några få).

--no-src **(eller *-S*)**

Hoppa över att kontakta Git-servern.

--no-build

Hoppa över byggprocessen.

--no-metadata

Ladda inte automatiskt ner den extra metadata som behövs för KDE:s git-moduler. Uppdateringen av själva modulernas källkod sker ändå om du inte också använder --no-src.

Detta kan vara användbart om du ofta kör om kdesrc-build, eftersom metadata inte ändras särskilt ofta. Observera dock att många andra funktioner kräver att metadata är tillgänglig. Du bör fundera på att köra kdesrc-build med väljaren --metadata-only en gång och därefter använda den här väljaren för efterföljande körningar.

--no-install

Installera inte automatiskt paket efter de har byggts.

--no-build-when-unchanged*, *--force-build

Det här alternativet inaktiverar explicit att hoppa över byggprocessen (en optimering styrd av alternativet build-when-unchanged). Det är användbart för att få kdesrc-build att utföra byggprocessen när någonting har ändrats som kdesrc-build inte kan kontrollera.

`--force-build` utför exakt samma funktion, och är kanske enklare att komma ihåg.

--debug

Aktiverar felsökningsläge för skriptet. För närvarande betyder det att all utmatning skickas till standardutmatningen förutom att loggas i loggkatalogen som vanligt. Dessutom är många funktioner mycket utförligare om vad de gör i felsökningsläge.

***--query*=typ**

Det här kommandot gör att kdesrc-build frågar efter en parameter i modulerna i bygglistan (antingen angivna på kommandoraden eller inlästa från inställningsfilen), och skriver ut resultatet på skärmen (en modul per rad).

Väljaren måste anges med en 'frågetyp', som ska vara en av följande:

- `source-dir`, vilket gör att kdesrc-build skriver ut den fullständiga sökvägen där modulens källkod är lagrad.
- `build-dir`, vilket gör att kdesrc-build skriver ut den fullständiga sökvägen där modulens byggprocess sker.
- `install-dir`, vilket gör att kdesrc-build skriver ut den fullständiga sökvägen där modulen installeras.
- `project-path`, vilket gör att kdesrc-build skriver ut modulens plats i hierarkin av KDE:s källkodsarkiv. Se Avsnitt 2.6.4 för mer information om denna hierarki.
- `branch`, vilket gör att kdesrc-build skriver ut den upplösta grenen i git som används för varje modul, baserat på gällande inställningar för tag, branch och branch-group.
- Annars kan namn som är giltiga för moduler i inställningsfilen användas som väljare, så listas det upplösta värdet för varje modul.

Om en enda modul anges på kommandoraden är utdata helt enkelt värdet på parametern som efterfrågas. Om flera (eller inga) moduler anges på kommandoraden, inleds varje rad med modulens namn. I båda fall slutar kdesrc-build att köra när värdena är utskrivna.

Väljaren lades till i kdesrc-build 16.05.

Exempelvis kommandot '**kdesrc-build** `--query branch kactivities kdepim`' kan sluta med följande utdata:

```
kactivities: master
kdepim: master
```

--refresh-build **(eller *-r*)**

Skapa om byggsystemet och bygg från grunden.

--reconfigure

Kör **cmake** (för KDE-moduler) eller **configure** (för Qt™) igen, utan att rensa byggkatalogen. Du ska normalt inte behöva ange detta, eftersom kdesrc-build detekterar när relevanta alternativ ändras och automatiskt kör om bygginställningen. Väljaren är implicit om `--refresh-build` används.

--resume-from **(eller *--from* eller *-f*)**

Alternativet är användbart för att återuppta byggprocessen från den angivna modulen, som ska vara nästa alternativ på kommandoraden. Man ska inte ange andra modulnamn på kommandoraden.

> **NOT**
> Alternativet lade tidigare till `--no-src`, men gör inte det längre (sedan kdesrc-build 1.13). Om du vill undvika uppdateringar av källkod vid återupptagande, skicka helt enkelt också med `--no-src` förutom övriga alternativ.

Se också: `--resume-after` (eller `--after` eller `-a`) och Avsnitt 6.3.6.1. Du bör föredra att använda den här kommandoradsväljaren om du har rättat byggfelet och vill att kdesrc-build ska göra färdigt byggprocessen.

***--resume-after* (eller *--after* eller *-a*)**

Alternativet används för att återuppta byggprocessen efter den angivna modulen, som ska vara nästa alternativ på kommandoraden. Man ska inte ange andra modulnamn på kommandoraden.

> **NOT**
> Alternativet lade tidigare till `--no-src`, men gör inte det längre (sedan kdesrc-build 1.13). Om du vill undvika uppdateringar av källkod vid återupptagande, skicka helt enkelt också med `--no-src` förutom övriga alternativ.

Se också: `--resume-from` (eller `--from` eller `-f`) och Avsnitt 6.3.6.1. Du bör föredra att använda den här kommandoradsväljaren om du har rättat byggfelet och har också byggt och installerat modulen själv, och vill att kdesrc-build ska starta igen med nästa modul.

--resume

Väljaren används för att köra kdesrc-build efter ett byggfel har uppstått.

Den återupptar bygget från modulen som misslyckades, med användning av listan över moduler som tidigare väntade på att byggas, och inaktiverar dessutom källkods- och metadatauppdateringar. När felet väl har rättats kan du snabbt komma tillbaka till att bygga modulerna som du tidigare byggde utan att mixtra med `--resume-from` och `--stop-before`.

Väljaren lades till i kdesrc-build 1.16.

***--stop-before* (eller *--until*)**

Kommandoradsväljaren används för att stoppa den normala byggprocessen precis *innan* en modul normalt skulle ha byggts.

Om den normala bygglistan exempelvis var modul-A, modul-B, modul-C, skulle `--stop-before=modul-B` göra att kdesrc-build bara bygger `modul-A`.

Kommandoradsväljaren lades till i kdesrc-build 1.16.

***--stop-after* (eller *--to*)**

Kommandoradsväljaren används för att stoppa den normala byggprocessen precis *efter* en modul normalt skulle ha byggts.

Om den normala bygglistan exempelvis var modul-A, modul-B, modul-C, skulle `--stop-after=modul-B` göra att kdesrc-build bygger `modul-A` och `modul-B`.

Kommandoradsväljaren lades till i kdesrc-build 1.16.

--stop-on-failure*, *--no-stop-on-failure

Väljaren gör att bygget avbryts så fort ett fel uppstår. Det normala beteendet är --stop-on-failure. Det går att överskrida om du vill fortsätta med återstående moduler i bygget, för att undvika att slösa tid ifall problemet gäller en enskild modul.

Alternativet lades till i kdesrc-build 1.16. Se också inställningsfilens alternativ stop-on-failure.

--rc-file

Det tolkar nästa kommandoradsväljare som filen att läsa konfigurationsinställningarna från. Förvalt värde för väljaren är `kdesrc-buildrc` (kontrolleras i arbetskatalogen). Om filen inte finns, används `~/.config/kdesrc-buildrc` (`$XDG_CONFIG_HOME/kdesrc-buildrc`, om `$XDG_CONFIG_HOME` är angivet) istället. Se också kapitel 4.

--print-modules

Utför alla åtgärder till och med beroendeomordning av modulerna angivna på kommandoraden (eller i inställningsfilen), skriver ut modulerna som skulle behandlats en per rad, och avslutar därefter utan ytterligare åtgärder.

Metadata för `kde-project` laddas först ner (se dock `--pretend` eller `--no-src`).

Utmatningen är inte fullständigt kompatibel med skriptanvändning, eftersom andra utmatningsmeddelanden kan genereras innan listan över moduler visas.

Detta är i huvudsak användbart för att snabbt avgöra vad kdesrc-build anser vara beroenden för en modul, vilket betyder att det bara är användbart för moduler i `kde-projects`. Väljaren är också kompatibel med `--resume-from`, `--resume-after`, `--stop-before` och `--stop-after`.

--list-build

Listar modulerna som skulle byggas, i den ordning de skulle byggas. Om tillämpligt, nämner listan också vilken incheckning, gren eller etikett som skulle väljas för utcheckning.

Väljaren liknar `--print-modules`. För mer information om hur moduler är relaterade till varandra, se också: `--dependency-tree`.

--dependency-tree

Skriver ut beroendeinformation för modulerna som skulle byggas genom att använda ett (rekursivt) trädformat. Den listade informationen omfattar också vilken specifik incheckning, gren eller etikett som de beror på, och om de beroende modulerna skulle byggas eller inte. Observera: Genererad utdata kan bli ganska omfattande för program med många beroenden.

--run

Alternativet tolkar nästa parameter på kommandoraden som ett program att köra, och kdesrc-build slutar då att läsa inställningsfilen, uppdaterar miljön som vanligt, och kör därefter angivet program.

Det fungerar dock inte för att starta ett skal med miljön från kdesrc-build i de flesta fall, eftersom interaktiva skal typiskt återställer åtminstone vissa av miljövariablerna (som `PATH` och `KDEDIRS`) under startsekvensen.

TIPS

Om du vill se miljön som används av kdesrc-build kan du köra kommandot **printenv**:

```
$ kdesrc-build --run printenv
KDE_SESSION_VERSION=4
SDL_AUDIODRIVER=alsa
LANGUAGE=
XCURSOR_THEME=Oxygen_Blue
LESS=-R -M --shift 5
QMAIL_CONTROLDIR=/var/qmail/control
... etc.
```

--prefix=</sökväg/till/kde>

Det här låter dig ändra katalogen som KDE installeras i från kommandoraden. Alternativet inbegriper *`--reconfigure`*, men det kan ändå krävas att *`--refresh-build`* används.

--revision

Väljaren gör att kdesrc-build checkar ut en särskilt numrerad version av varje Git-modul, och överskrider eventuella alternativ som branch, tag eller revision redan är angivna för modulerna.

Väljaren är troligen inte en god idé, och stöds bara för kompatibilitet med äldre skript.

--build-system-only

Väljaren gör att kdesrc-build avbryter bygga en modul precis innan kommandot **make** skulle ha körts. Det stöds bara för kompatibilitet med äldre versioner, effekten är inte till hjälp för det nuvarande byggsystemet för KDE.

--install

Om detta är den enda kommandoradsväljaren, försöker det installera alla moduler som finns i `log/latest/build-status`. Om kommandoradsväljare anges efter `--install`, antas de alla vara moduler att installera (även om de inte byggdes med lyckat resultat vid den senaste körningen).

--no-snapshots

Att ange den här väljaren gör att kdesrc-build alltid utför en normal ursprunglig utcheckning av en modul istället för att använda en sparad version för snabbstart (bara tillgängligt för Git-moduler från arkivet `kde-projects`). Observera att alternativet bör bara användas om ett fel uppstår vid användning av en sparad version, eftersom sparade versioner för snabbstart reducerar belastningen på KDE:s källkodsarkiv.

> NOT
> Sparade versioner av modulen *är* riktiga utcheckningar. Du ska inte behöva ange den här väljaren, den är bara till hjälp vid felsökning.

--delete-my-patches

Väljaren används för att låta kdesrc-build ta bort källkataloger som kan innehålla användardata, så att modulerna kan laddas ner igen. Den är normalt bara användbar för KDE-utvecklare (som kan ha lokala ändringar som skulle tas bort).

Normalt ska inte väljaren användas, utan kdesrc-build ber att få köras om med den vid behov.

--delete-my-settings

Väljaren används för att låta kdesrc-build skriva över befintliga filer som kan innehålla användardata.

För närvarande används den bara för inställning av en xsession för inloggningshanteraren. Normalt ska inte väljaren användas, utan kdesrc-build ber att få köras om med den vid behov.

--<alternativnamn>=

Du kan använda alternativet för att överskrida ett alternativ i inställningsfilen för alla moduler. För att till exempel överskrida alternativet log-dir, skulle du skriva **--log-dir=/sökväg/till/katalog**.

> NOT
> Funktionen kan bara användas för alternativnamn som redan känns igen av kdesrc-build som ännu inte stöds av relevanta kommandoradsväljare. Exempelvis har inställningsfilens alternativ async de specifika kommandoradsväljarna --async och --no-async som föredras av kdesrc-build.

--set-module-option-value=<modulnamn>,<alternativnamn>,<alternativvärde>

Du kan använda alternativet för att överskrida ett alternativ i inställningsfilen för en specifik modul.

Alla övriga kommandoradsväljare antas vara moduler att uppdatera och bygga. Blanda helst inte ihop bygga och installera.

Kapitel 6

Använda kdesrc-build

6.1 Förord

Efter du har gått igenom kapitel 2, är det normalt så enkelt att använda kdesrc-build som att skriva följande från en terminal:

```
% kdesrc-build
```

kdesrc-build laddar därefter ner KDE:s källkod, försöker att konfigurera och bygga den, och därefter installera den.

Läs vidare för att upptäcka hur kdesrc-build gör detta, och vad du mer kan göra med verktyget.

6.2 Grundläggande funktioner i kdesrc-build

6.2.1 stöd för qt

kdesrc-build stöder att bygga Qt™-verktygslådan som används an KDE-programvara som en bekvämlighet för användaren. Stödet hanteras av en särskild modul som kallas qt.

> NOT
> Qt™ utvecklas i ett separat arkiv från KDE-programvara, lokaliserat på http://code.qt.io/cgit/qt/ .

För att bygga Qt™ måste du försäkra dig om att inställningen qtdir pekar på katalogen där du vill att Qt™ ska installeras, som beskrivs i Avsnitt 2.2.

Därefter måste du försäkra dig om att modulen qt läggs till i `kdesrc-buildrc`, innan några andra moduler i filen. Om du använder exemplet på inställningsfil, kan du helt enkelt ta bort kommentarerna från den befintliga posten för modulen qt.

Nu bör du kontrollera att alternativet repository och alternativen branch är lämpligt inställda:

1. Det första möjligheten är att bygga Qt™ med användning av en spegelplats som underhålls i KDE:s källkodsarkiv (inga andra ändringar har lagts till, det är helt enkelt en klon av den officiella källan). Det rekommenderas starkt på grund av att det ibland uppstår problem med att klona hela Qt™-modulen från det officiella arkivet.

   Alternativet `repository` kan ställas in till **`kde:qt`** för modulen qt, för att använda den möjligheten.

2. Annars, för att bygga standard-Qt™, ställ in alternativet `repository` till **`git://gitorious.org/qt/qt.git`**. Observera att det kan uppstå problem att skapa den ursprungliga klonen av Qt™ från detta arkiv.

I båda fall ska alternativet branch ställas in till **`master`** (om du inte vill bygga en annan gren).

6.2.2 Standardflaggor tillagda av kdesrc-build

Observera: avsnittet gäller inte för moduler där du har ställt in en egen verktygskedja, t.ex. genom att använda cmake-toolchain.

För att spara tid, lägger kdesrc-build till några standardsökvägar i miljön åt dig:

- Sökvägen till KDE- och Qt™-biblioteken läggs till i variabeln `LD_LIBRARY_PATH` automatiskt. Det betyder att du inte behöver redigera libpath för att inkludera dem.
- Sökvägen till program för utvecklingsstöd i KDE och Qt™ läggs till i variabeln `PATH` automatiskt. Det betyder att du inte behöver redigera binpath för att inkludera dem.
- Sökvägen till pkg-config som tillhandahålls av KDE läggs till i variabeln `PKG_CONFIG_PATH` automatiskt. Det betyder att du inte behöver använda set-env för att lägga till dem.
- Inställningen av kdedir skickas automatiskt vidare till miljövariabeln `KDEDIR` under byggprocessen. (`KDEDIRS` påverkas inte).
- Inställningen av qtdir skickas automatiskt vidare till miljövariabeln `QTDIR` under byggprocessen.

6.2.3 Ändra byggprioritet i kdesrc-build

Program kan köra med olika prioritetsnivåer på operativsystem, inklusive Linux® och BSD. Det tillåter systemet att tilldela tid för de olika programmen enligt hur viktiga de är.

kdesrc-build tilldelar normalt sig själv låg prioritet så att resten av programmen på systemet är opåverkade och kan köra normalt. Genom att använda den här tekniken, använder kdesrc-build extra processorkraft när den är tillgänglig.

kdesrc-build behåller fortfarande prioritetsnivån nog hög så att det kör innan rutinmässiga bakgrundsprocesser och innan program som donerar processorkraft såsom Seti@Home.

För att ändra kdesrc-build så att det använder en högre (eller lägre) prioritetsnivå permanent, måste du justera inställningen av niceness i inställningsfilen. Alternativet niceness styr hur 'snällt' kdesrc-build är mot andra program. Med andra ord, att ha en högre niceness ger kdesrc-build lägre prioritet. Så för att ge kdesrc-build en högre prioritet, reducera niceness (och tvärtom). niceness kan gå från 0 (inte alls snällt, högsta prioritet) till 20 (supersnällt, lägsta prioritet).

Du kan också tillfälligt ändra prioritet hos kdesrc-build genom att använda kommandoradsväljaren --nice. Väljarens värde är exakt samma som för niceness.

> **NOT**
> Det är möjligt för vissa program som körs av systemadministratören att ha ett negativt snällhetsvärde, vilket motsvarar ännu högre prioritet för sådana program. Att ange en negativ niceness (eller till och med 0) för kdesrc-build är inte en god idé, eftersom det inte förbättrar körtiden nämnvärt, men gör att datorn verkar mycket slö om du ändå behöver använda den.

För att köra kdesrc-build med snällhetsgrad 15 (en lägre prioritet än normal) skriv:

```
% kdesrc-build --nice=15
```

Eller kan du redigera inställningsfilen för att göra ändringen permanent:

```
niceness 15
```

> **TIPS**
> Alternativet niceness påverkar bara användning av datorns processor(er). En annan stor effekt på datorns prestanda har att göra med hur mycket in- och utmatning av data (I/O) som ett program använder. För att bestämma hur mycket I/O ett program kan använda, stöder moderna Linux®-operativsystem ett liknande verktyg som kallas ionice. kdesrc-build stöder ionice (men bara för att aktivera eller inaktivera det helt och hållet), med alternativet use-idle-io-priority, sedan kdesrc-build version 1.12.

6.2.4 Installera som systemadministratör

Du kanske också vill att kdesrc-build ska köra installationen med rättigheter som systemadministratör. Det kan gälla för systeminstallation, som inte rekommenderas. Det är dock också användbart när den rekommenderade installationen som enanvändare av KDE används. Det beror på att vissa moduler (i synnerhet kdebase) installerar program som tillfälligt behöver förhöjda rättigheter när de kör. De kan inte uppnå dessa rättighetsnivåer om de inte installeras med förhöjda rättigheter.

Du skulle helt enkelt kunna köra kdesrc-build som systemadministratör direkt, men det rekommenderas inte, eftersom programmet inte har granskats för den sortens användning. Även om det bör vara säkert att köra programmet på det sättet, är det bättre att undvika att köra som systemadministratör när det är möjligt.

För att hantera detta, tillhandahåller kdesrc-build alternativet make-install-prefix. Du kan använda alternativet för att ange ett kommando som används för att utföra installationen som en annan användare. Det rekommenderade sättet att använda kommandot är med programmet Sudo, som kör installationskommandot som systemadministratör.

För att till exempel installera alla moduler genom att använda Sudo, skulle du kunna göra något liknande:

```
global
  make-install-prefix sudo
  # Övriga alternativ
end global
```

För att använda make-install-prefix för bara en enda modul, skulle det här fungera:

```
module något-modul-namn
  make-install-prefix sudo
end module
```

6.2.5 Visa förloppet för en byggprocess av en modul

Funktionen är alltid tillgänglig, och automatiskt aktiverad om möjligt. Vad den gör är att visa ett uppskattat byggförlopp medan en modul byggs, på så sätt vet du hur mycket längre det kommer att ta att bygga en modul.

6.3 Avancerade funktioner

6.3.1 Delvis bygga en modul

Det är möjligt att bara bygga delar av en enda KDE-modul. Du kanske till exempel bara vill kompilera ett program från en modul. kdesrc-build har funktioner som gör det enkelt. Det finns flera komplementära sätt att göra det.

6.3.1.1 Ta bort kataloger från en byggplats

Det möjligt att ladda ner ett helt arkiv men låta byggsystemet utelämna ett antal kataloger när byggprocessen utförs. Det kräver att modulen använder CMake och att modulens byggsystem tillåter att katalogen som ska tas bort är valfri.

Det bestäms med alternativet do-not-compile.

> VIKTIGT
> Alternativet kräver minst att byggsystemet för modulen konfigureras om efter det har ändrats. Det görs med kommandot **`kdesrc-build --reconfigure modul`**.

För att ta bort katalogen `python` från byggprocessen för kdebindings:

```
module kdebindings
  do-not-compile python
end module
```

> NOT
> Funktionen beror på vissa standardkonventioner som används i de flesta moduler i KDE. Därför kanske den inte fungerar för alla program.

6.3.2 Stöd för grenar och taggar i kdesrc-build

6.3.2.1 Vad är grenar och taggar?

Git stöder hantering av historik för KDE:s källkod. KDE använder stödet för att skapa grenar för utveckling, och att ge arkivet en tagg då och då med utgåvan av en ny version.

Till exempel kan utvecklarna av KMail arbeta med en ny funktion i en annan gren för att undvika att förstöra versionen som används av de flesta utvecklare. Grenen har pågående utveckling, samtidigt som huvudgrenen (som kallas master) kan ha pågående utveckling.

En tagg är å andra sidan en ögonblicksbild av källkodsarkivet vid en viss tid. Det används av KDE:s administrationsgrupp för att markera en version av koden som är lämplig för en utgåva och fortfarande tillåta att utvecklare arbetar med koden.

6.3.2.2 Hur man använder grenar och taggar

Stöd för grenar och taggar hanteras med en uppsättning alternativ, som rör sig från en generell begäran om en version, till en specifik webbadress att ladda ner för avancerade användare.

Det enklaste sättet är att använda alternativen branch och tag. Du använder helt enkelt alternativet tillsammans med namnet på den önskade grenen eller taggen för en modul, så försöker

kdesrc-build avgöra lämplig plats i KDE:s arkiv att ladda ner från. För de flesta moduler i KDE fungerar det mycket bra.

För att ladda ner kdelibs från KDE 4.6 (som helt enkelt är känt som grenen 4.6):

```
module kdelibs
  branch 4.6
  # övriga alternativ...
end module
```

Eller, för att ladda ner kdemultimedia som det var när det gavs ut med KDE 4.6.1:

```
module kdemultimedia
  tag 4.6.1
  # övriga alternativ...
end module
```

> **TIPS**
> Du kan ställa in ett allmänt grenvärde, men om du gör det, glöm inte att ange en annan gren för moduler som inte ska använda den allmänna grenen.

6.3.2.3 Stöd för avancerade grenalternativ

kdesrc-build stöder två alternativ i situationer då branch och tag gissar fel angående den riktiga sökvägen: module-base-path och override-url.

- Alternativet module-base-path används för att hjälpa kdesrc-build fylla i saknade delar av modulens sökväg. I KDE:s arkiv är alla sökvägar av formen `gitRoot/module-base-path/ modulnamn`. Normalt kan kdesrc-build räkna ut den lämpliga delen i mitten själv. När det inte går, kan du använda module-base-path så här:

  ```
  module kdesupport
      # kdesupport stöder diverse taggar för att enkelt organisera
      # programvaran för en given utgåva av KDE-plattformen.
      module-base-path tags/kdesupport-for-4.5
  end module
  ```

  Det skulle göra att kdesrc-build laddar ner kdesupport från (i det här exemplet) `git://invent.kde.org/home/kdetags/kdesupport-for-4.5`.

  > **TIPS**
  > I tidigare versioner av kdesrc-build, hanterades module-base-path annorlunda. Om du stöter på problem vid användning av en gammal definition av module-base-path, bör du nog verifiera att den verkliga sökvägen är den som kdesrc-build förväntar sig genom att använda väljaren --pretend.

- Alternativet override-url kräver å andra sidan att du anger den exakta sökvägen att ladda ner från. Det låter dig dock hämta från sökvägar som kdesrc-build inte skulle ha någon möjlighet att ladda ner från. För närvarande bör alternativet module-base-path vara tillräckligt för alla webbadresser för Git källkod.

  > **VIKTIGT**
  > kdesrc-build rör inte eller korrigerar något värde som du anger för override-url överhuvudtaget, så om du byter inställningen repository, kan du också behöva uppdatera det här.

6.3.3 Stoppar bygget i förtid

6.3.3.1 Bygget fortsätter normalt även om allvarliga fel uppstår

kdesrc-build uppdaterar, bygger och installerar normalt alla moduler i den angivna listan över moduler att bygga, även om en modul misslyckas att bygga. Detta är vanligtvis en bekvämlighet för att du ska kunna uppdatera programvarupaket även om ett enkelt misstag görs i ett av källkodsarkiven under utvecklingen som gör att bygget fallerar.

Du kan dock önska att kdesrc-build stoppar vad det gör när det inte lyckas att bygga och installera en modul. Det kan hjälpa dig att spara tid som kommer att gå till spillo på att komma vidare när moduler som finns kvar i bygglistan inte heller kommer att kunna byggas med lyckat resultat, särskilt om du aldrig har lyckats bygga modulerna i listan.

6.3.3.2 Stoppar inte i förtid med --no-stop-on-failure

Den primära metoden för att göra det är att använda kommandoradsväljaren --no-stop-on-failure när kdesrc-build körs.

Väljaren kan också anges i inställningsfilen för att göra den till normalt beteende.

Det är också möjligt att tala om för kdesrc-build att sluta bygga under körning *efter* att ha slutfört den aktuella modulen den arbetar med. Det är i motsats till att avbryta kdesrc-build med ett kommando som **Ctrl**+**C**, vilket avbryter kdesrc-build omedelbart, och förlorar arbetet för den aktuella modulen.

> VIKTIGT
> Avbryta kdesrc-build under installation av en module när alternativet use-clean-install är aktiverat innebär att den avbrutna modulen är otillgänglig tills kdesrc-build kan bygga modulen med lyckat resultat. Om du behöver avbryta kdesrc-build utan att tillåta en snygg avstängning i detta fall, försök åtminstone undvika att göra detta medan kdesrc-build installerar en modul.

6.3.3.3 Stoppar kdesrc-build snyggt när stop-on-failure är false

Som nämnts ovan är det möjligt att få kdesrc-build att snyggt avsluta tidigt när det väl har slutfört modulen det arbetar med. För att göra det måste du skicka signalen POSIX `HUP` till kdesrc-build

Du kan göra detta med ett kommando som **pkill** (på Linux®-system) enligt följande:

```
$ pkill -HUP kdesrc-build
```

Om det lyckas kommer du att se ett meddelande i utmatningen från kdesrc-build som liknar:

```
[ build ] tog emot SIGHUP, kommer att avsluta efter den här modulen
```

> NOT
> kdesrc-build kan visa meddelandet flera gånger beroende på antalet individuella kdesrc-build processer som är aktiva. Det är normalt och inte en indikation av ett fel.

När kdesrc-build har bekräftat signalen kommer den att sluta bearbetningen efter att den aktuella modulen har byggts och installerats. Om kdesrc-build fortfarande uppdaterar källkoden när begäran tas emot, kommer kdesrc-build att stoppa efter uppdateringen av modulens källkod är klar. När både uppdaterings- och byggprocessen har stoppats tidigt, kommer kdesrc-build skriva ut sina delresultat och avsluta.

6.3.4 Hur kdesrc-build försöker försäkra sig om en lyckad byggprocess

6.3.4.1 Automatisk ombyggnad

kdesrc-build innehöll tidigare funktioner för att automatiskt försöka bygga om modulen efter ett fel (eftersom ibland fungerade försöket att göra om, på grund av fel i byggsystemet vid den tiden). Tack vare bytet till CMake lider inte byggsystemet längre av dessa fel, och därför försöker inte kdesrc-build bygga en modul mer än en gång. Det finns dock situationer då kdesrc-build automatiskt utför åtgärder.

- Om du ändrar configure-flags eller cmake-options för en modul, detekterar kdesrc-build det och kör automatiskt om configure eller cmake för modulen.
- Om byggsystemet inte finns (även om kdesrc-build inte tog bort det) skapar kdesrc-build automatiskt om det. Det är användbart för att ge möjligheten att utföra en fullständig --refresh-build för en specifik modul utan att den utförs för andra moduler.

6.3.4.2 Bygga om en modul manuellt

Om du gör en ändring i en moduls inställningsalternativ, eller om modulens källkod ändras på ett sätt som kdesrc-build inte känner igen, kan du behöva bygga om modulen manuellt.

Du kan göra det genom att helt enkelt köra **`kdesrc-build --refresh-build modul`**.

Om du skulle vilja att kdesrc-build istället automatiskt bygger om modulen under nästa normala bygguppdatering, kan du skapa en särskild fil. Varje modul har en byggkatalog. Om du skapar en fil som heter `.refresh-me` i en moduls byggkatalog, bygger kdesrc-build om modulen nästa gång byggprocessen sker, även om det normalt skulle utföra den snabbare inkrementella byggprocessen.

> **TIPS**
> Normalt är byggkatalogen `~/kdesrc/build/ modul /` . Om du ändrar inställning av alternativet build-dir, använd den istället för `~/kdesrc/build`.

Bygg om med `.refresh-me` för modulen *`kdelibs`*:

```
% touch ~/kdesrc/build/kdelibs/.refresh-me
% kdesrc-build
```

6.3.5 Ändra inställning av miljövariabler

Normalt använder kdesrc-build miljön som är närvarande vid start när program körs för att utföra uppdateringar och bygga. Det är användbart när du kör kdesrc-build från kommandoraden.

Dock kan du vilja ändra inställning av miljövariabler som kdesrc-build inte direkt tillhandahåller ett alternativ för. (För att exempelvis ställa in eventuella miljövariabler som behövs när kdesrc-build körs i bakgrunden via Cron.) Det är möjligt med alternativet set-env.

I motsats till de flesta alternativ kan det anges flera gånger, och accepterar två värden, åtskilda med mellanslag. Det första är namnet på miljövariabeln som ska sättas, och resten av raden är dess värde.

Ställ in **`DISTRO=BSD`** för alla moduler:

```
global
  set-env DISTRO BSD
end global
```

6.3.6 Återuppta byggprocesser

6.3.6.1 Återuppta en misslyckad eller avbruten byggprocess

Du kan tala om för kdesrc-build att börja bygga från en annan modul än det normalt skulle göra. Det kan vara användbart när en uppsättning moduler misslyckades, eller om du avbröt en körning i mitten. Du kan styra det med väljarna --resume-from och --resume-after.

> NOT
> Äldre versioner av kdesrc-build hoppade över uppdatering av källkod när ett bygge återupptogs. Det görs inte längre standardmässigt, men man kan alltid använda kommandoradsväljaren `--no-src` för att hoppa över uppdateringen.

Återuppta byggprocessen med början på kdebase:

```
% kdesrc-build --resume-from=kdebase
```

Återuppta byggprocessen med början efter kdebase (i fallet du fixade problemet för hand och installerade modulen själv):

```
% kdesrc-build --resume-after=kdebase
```

Om det senaste bygget med kdesrc-build slutade med ett byggfel, kan du också använda kommandoradsväljaren --resume, som återupptar det senaste bygget med modulen som misslyckades. Källkods- och metadatauppdateringar hoppas också över (men om du behöver dem, är det i allmänhet bättre att använda --resume-from istället).

6.3.6.2 Ignorera moduler i en byggprocess

På samma sätt som du kan återuppta byggprocessen från en modul, kan du istället välja att uppdatera och bygga normalt, men ignorera en uppsättning moduler.

Du kan göra det med väljaren --ignore-modules. Den talar om för kdesrc-build att ignorera alla följande moduler på kommandoraden när en uppdatering och byggprocess utförs.

Ignorera extragear/multimedia och kdereview under en fullständig körning:

```
% kdesrc-build --ignore-modules extragear/multimedia kdereview
```

6.3.7 Ändra alternativ från kommandoraden

6.3.7.1 Ändra allmänna alternativ

Du kan ändra uppsättningen alternativ som läses från inställningsfilen direkt från kommandoraden. Ändringen överskrider inställningen i filen, men är bara tillfällig. Den gäller bara så länge den fortfarande är närvarande på kommandoraden.

kdesrc-build låter dig ändra alternativ namngivna som *`alternativnamn`* genom att ange en väljare på kommandoraden på formen **`--alternativnamn=värde`**. kdesrc-build känner igen om det inte vet vad alternativet är, ock söker efter namnet i sin lista med alternativnamn. Om det inte känner igen namnet, får du en varning, annars kommer det ihåg värdet du ställde in det till och överskrider en eventuell inställning från filen.

Ställa in alternativet source-dir till `/dev/null` för test:

```
% kdesrc-build --pretend --källkodskatalog=/dev/null
```

6.3.7.2 Ändra modulalternativ

Det är också möjligt att bara ändra alternativ för en viss modul. Syntaxen är liknande: `--modul,alternativnamn=värde`.

Ändringen överskrider eventuell duplicerad inställning för modulen som hittas i inställningsfilen, och gäller bara när alternativet skickas med på kommandoraden.

Använda en annan byggkatalog för modulen kdeedu:

```
% kdesrc-build --kdeedu,byggkatalog=temporärbygg
```

6.4 Funktioner för KDE-utvecklare

6.4.1 Kontroll av SSH-agent

kdesrc-build kan försäkra sig om att KDE-utvecklare som använder SSH för att komma åt KDE:s källkodsarkiv inte av misstag glömmer att lämna verktyget SSH-agenten aktiverad. Det kan orsaka att kdesrc-build hänger sig för alltid medan det väntar på att utvecklaren ska skriva in lösenordet till SSH, så normalt kontrollerar kdesrc-build att agenten kör innan det utför uppdateringar av källkoden.

> NOT
> Det görs bara för KDE-utvecklare som användare SSH.

Du kanske vill inaktivera kontrollen av SSH-agenten, i situationer där kdesrc-build detektera närvaron av en agent av misstag. För att inaktivera kontroll av agenten, sätt alternativet `disable-agent-check` till **`true`**.

Inaktivera SSH-agentkontroll:

```
global
  disable-agent-check true
end global
```

6.5 Andra funktioner i kdesrc-build

6.5.1 Ändra mängden utmatning från kdesrc-build

kdesrc-build har flera väljare för att styra mängden utmatning som skriptet skapar. Hur som helst, kommer fel alltid att matas ut.

- Väljaren `--quiet` (den korta formen är `-q`) gör att kdesrc-build i huvudsak är tyst. Bara viktiga meddelanden, varningar eller fel visas. Om tillgängligt visas fortfarande förloppsinformation för byggprocessen.
- Väljaren `--really-quiet` (ingen kortform) gör att kdesrc-build bara visar viktiga varningar eller fel när det kör.
- Väljaren `--verbose` (kortformen är `-v`) gör att kdesrc-build är mycket detaljerad i utmatningen.
- Väljaren `--debug` är bara till för felsökningssyfte. Den gör att kdesrc-build beter sig som om `--verbose` är aktiverad, gör också att kommandon matas ut på terminalen, och visar felsökningsinformation för många funktioner.

6.5.2 Färgutmatning

När kdesrc-build körs från Konsole eller en annan terminal, visas normalt färglagd text.

Du kan inaktivera det genom att använda väljaren `--no-color` på kommandoraden, eller ställa in alternativet colorful-output i inställningsfilen till **`false`**.

Inaktivera färgutmatning i inställningsfilen:

```
global
  colorful-output false
end global
```

6.5.3 Ta bort onödiga kataloger efter en byggprocess

Har du ont om diskutrymme men vill ändå köra de allra senaste utcheckningen av KDE? kdesrc-build kan hjälpa dig att reducera diskanvändning medan du bygger KDE från Git.

> NOT
> Var medveten om att det går åt mycket utrymme för att bygga KDE. Det finns flera stora delar som använder utrymme när kdesrc-build används:

1. Själva utcheckningen av källkoden kan uppta ett försvarligt utrymme. Standardmodulerna upptar ungefär 1.6 Gibyte diskutrymme. Du kan reducera storleken genom att försäkra dig om att du bara bygger så många moduler som du verkligen vill ha. kdesrc-build tar inte bort källkod från disken även om du tar bort posten från inställningsfilen, så försäkra dig om att du går till och tar bort oanvända utcheckningar i källkodskatalogen. Observera att källkodsfilerna laddas ner från Internet, och du *ska inte* ta bort dem om du faktiskt använder dem, åtminstone till du är klar med användning av kdesrc-build.

   Om du dessutom har installerat Qt™ från distributionen (och chansen är stor att du har det), behöver du troligen inte installera modulen qt. Det hyvlar av omkring 200 Mibyte från källkodsstorleken på disk.

2. kdesrc-build skapar en separat byggkatalog för att bygga källkoden. Ibland måste kdesrc-build kopiera en källkodskatalog för att skapa en falsk byggkatalog. När det sker, används platsbesparande symboliska länkar, så det bör inte vara något krångel med diskutrymme. Byggkatalogen är typiskt mycket större än en moduls källkodskatalog. Byggkatalogen för kdebase är till exempel omkring 1050 Mibyte, medan källkoden för kdebase bara är omkring 550 Mibyte.

   Som tur är krävs inte byggkatalogen efter en modul har byggts och installerats med lyckat resultat. kdesrc-build kan automatiskt ta bort byggkatalogen efter en modul har installerats. Se exemplen nedan för mer information. Observera att genom att utföra detta steg, blir det omöjligt för kdesrc-build att utföra tidsbesparande inkrementella byggprocesser.

3. Till sist krävs diskutrymme för själva installationen av KDE, som inte körs från byggkatalogen. Det upptar typiskt mindre utrymme än byggkatalogen. Det är dock svårare att få fram exakta siffror.

Hur reducerar man utrymmeskraven för KDE? Ett sätt är att använda riktiga kompilatorflaggor för att optimera för reducering av utrymme istället för hastighet. Ett annat sätt, som kan ha stor effekt, är att ta bort felsökningsinformation från det färdigbyggda KDE.

> **VARNING**
> Du bör vara mycket säker på att du vet vad du gör innan du bestämmer dig för att ta bort felsökningsinformation. Att köra den allra senaste programvaran betyder att du kör programvara som potentiellt har mycket större sannolikhet att krascha än stabila utgåvor. Om du kör programvara utan felsökningsinformation kan det vara mycket svårt att skapa en bra felrapport för att få problemet löst, och du måste troligen aktivera felsökning igen för programmet som påverkas och bygga om för att hjälpa en utvecklare att rätta kraschen. Så ta bort felsökningsinformation på egen risk!

Ta bort byggkatalogen efter en modul har installerats. Källkoden behålls fortfarande, och felsökning är aktiverad:

```
global
  configure-flags      --enable-debug
  remove-after-install builddir        # Ta bort byggkatalog efter  ↩
      installation
end global
```

Ta bort byggkatalogen efter installation, utan felsökningsinformation, med storleksoptimering.

```
global
  cxxflags             -Os             # Optimera för storlek
  configure-flags      --disable-debug
  remove-after-install builddir        # Ta bort byggkatalog efter  ↩
      installation
end global
```

Kapitel 7

CMake, byggsystemet för KDE

7.1 Introduktion till CMake

Under mars 2006, slog programmet CMake ut flera konkurrenter och valdes som byggsystem för KDE 4, och ersätter systemet baserat på autotools som KDE hade använt från början.

En sida med en introduktion till CMake är tillgänglig på KDE Community Wiki. Istället för att köra **`make -f Makefile.cvs`**, därefter **configure** och sedan Make, kör vi helt enkelt CMake och därefter Make.

kdesrc-build har initialt stöd för CMake. Några funktioner i kdesrc-build var i själva verket funktioner i det underliggande byggsystemet, inklusive configure-flags och do-not-compile. När motsvarande funktioner är tillgängliga, tillhandahålls de. Motsvarigheten till alternativet configure-flags är till exempel cmake-options, och alternativet do-not-compile stöds också för CMake från och med kdesrc-build version 1.6.3.

Kapitel 8

Tack till och licens

Översättning Stefan Asserhäll stefan.asserhall@bredband.net

Bilaga A

KDE-moduler och organisation av källkoden

A.1 'Modulen'

KDE grupperar programvaran i 'moduler' av olika storlek. Det var från början en lös gruppering av några få stora moduler, men vid introduktionen av Git-baserade källkodsarkiv, delades dessa stora moduler ytterligare i många mindre moduler.

kdesrc-build använder också modulkonceptet. I stort sett är en 'modul' en gruppering av kod som kan laddas ner, byggas, testas och installeras.

A.1.1 Enskilda moduler

Det är enkelt att ställa in kdesrc-build att bygga en enstaka modul. Följande listning är ett exempel på hur en deklaration för en modul baserad på Git skulle se ut i inställningsfilen.

```
module kdexxx
    cmake-options -DCMAKE_BUILD_TYPE=Debug
end module
```

> TIPS
> Det är en modul baserad på Git eftersom den inte använder alternativet repository. Dessutom listas alternativet `cmake-options` bara som ett exempel, det krävs inte.

A.1.2 Grupper av relaterade moduler

Nu är de flesta KDE-moduler baserade på Git, och kombineras normalt i modulgrupper.

Därför stöder kdesrc-build också modulgrupper, med module sets. Ett exempel:

```
module-set base-modules
    repository kde-projects
    use-modules kde-runtime kde-workspace kde-baseapps
end module-set
```

> **TIPS**
> Du kan lämna moduluppsättningens namn tomt om du vill (*base-modules* i detta fall). Inställningen `repository` talar om för kdesrc-build varifrån källkoden ska laddas ner, men du kan också använda en webbadress som börjar med git://.

En särskild funktion med '`repository kde-projects`' är att kdesrc-build automatiskt inkluderar alla Git-moduler som är grupperade under modulerna du listar (i KDE:s projekdatabas).

A.1.3 Modulen 'branch groups'

När konceptet med en grupp av moduler fördes vidare, fann KDE-utvecklarna till slut att synkronisering av namnen på Git-grenarna över ett stort antal arkiv började bli svårt, särskilt under utvecklingsoffensiven av det nya KDE-ramverket för Qt™ 5.

Alltså utvecklades konceptet 'branch groups' (grengrupper) för att låta användare bara välja en eller några få grupper och låta skriptet automatiskt välja lämplig Git-gren.

kdesrc-build stöder funktionen från version 1.16-pre2, via alternativet branch-group.

Example A.1 Exempel på användning av branch-group

branch-group kan användas på följande sätt i inställningsfilen:

```
global
    # Välj KDE Frameworks 5 och andra Qt5-baserade program
    branch-group kf5-qt5

    # Andra globala alternativ här ...
end global

module-set
    # branch-group fungerar bara för kde-projects
    repository kde-projects

    # branch-group ärvs från den som ställs in globalt, men skulle kunna
    # specificeras här.

    use-modules kdelibs kde-workspace
end module-set

# Gren för kdelibs kommer att vara "frameworks"
# Gren för kde-workspace kommer att vara "master" (från augusti 2013)
```

I detta fall ger samma `branch-group` som ger olika namn på grenar för varje Git-modul.

Funktionen kräver att en del data underhålls av KDE-utvecklarna i Git-arkivet vid namn `kde-build-metadata`. Dock inkluderas modulen automatiskt av kdesrc-build (även om du kan se den visas i skriptets utdata).

> **TIPS**
> För KDE-moduler som inte har ett inställt namn på grenen för grengruppen du väljer, får normalt ett lämpligt grennamn, som om du inte hade angivit `branch-group` alls.

Bilaga B

Ersatta procedurer för att ställa in en profil

B.1 Ställa in en inloggningsprofil för KDE

Instruktionerna täcker hur man ställer in profilen som krävs för att säkerställa att datorn kan logga in på det nybyggda KDE Plasma-skrivbordet. kdesrc-build försöker normalt göra det automatiskt (se Avsnitt 2.5.1). Detta appendix kan vara användbart för de som inte kan använda det inbyggda stödet i kdesrc-build för inställning av inloggningsprofil. Dock kanske instruktionerna inte alltid är aktuella, och det kan också vara användbart att titta i filen `kde-env-master.sh` som ingår i källkoden för kdesrc-build.

B.1.1 Ändra startprofilinställningar

> VIKTIGT
> Filen `.bash_profile` är inloggningsinställningarna för det populära skalet bash som används av många Linux®-distributioner. Om du använder ett annat skal, kan du behöva justera exemplen som ges i det här avsnittet för ditt specifika skal.

Öppna eller skapa filen `.bash_profile` i hemkatalogen med din favoriteditor, och lägg till följande i slutet på filen. Om du bygger modulen qt (det gör du normalt), lägg istället till:

```
QTDIR=(sökväg till qtdir)   # Såsom normalt ~/kdesrc/build/qt.
KDEDIR=(sökväg till kdedir) # Såsom normalt ~/kde.
KDEDIRS=$KDEDIR
PATH=$KDEDIR/bin:$QTDIR/bin:$PATH
MANPATH=$QTDIR/doc/man:$MANPATH

# Gör rätt om inte LD_LIBRARY_PATH redan är satt.
if [ -z $LD_LIBRARY_PATH ]; then
  LD_LIBRARY_PATH=$KDEDIR/lib:$QTDIR/lib
else
  LD_LIBRARY_PATH=$KDEDIR/lib:$QTDIR/lib:$LD_LIBRARY_PATH
fi

export QTDIR KDEDIRS PATH MANPATH LD_LIBRARY_PATH
```

eller om du inte bygger qt (och använder systemets Qt™) lägg till det här istället:

```
KDEDIR=(sökväg till kdedir)  # Såsom ~/kde normalt.
KDEDIRS=$KDEDIR
PATH=$KDEDIR/bin:$QTDIR/bin:$PATH

# Gör rätt om inte LD_LIBRARY_PATH redan är satt.
if [ -z $LD_LIBRARY_PATH ]; then
  LD_LIBRARY_PATH=$KDEDIR/lib
else
  LD_LIBRARY_PATH=$KDEDIR/lib:$LD_LIBRARY_PATH
fi

export KDEDIRS PATH LD_LIBRARY_PATH
```

Om du inte använder en särskild användare ställ in en annan $KDEHOME för din nya miljö i .bash_profile:

```
export KDEHOME="${HOME}/.kde-git"

# Create it if needed
[ ! -e ~/.kde-git ] && mkdir ~/.kde-git
```

NOT

Om din K-meny senare är tom eller för tätpackad med program från din distribution, kanske du måste ange miljövariabeln XDG i din .bash_profile:

```
XDG_CONFIG_DIRS="/etc/xdg"
XDG_DATA_DIRS="${KDEDIR}/share:/usr/share"
export XDG_CONFIG_DIRS XDG_DATA_DIRS
```

B.1.2 Starta KDE

När du nu har justerat dina miljöinställningar för att använda rätt KDE, är det viktigt att försäkra dig om att det rätta **startkde**-skriptet också används.

Öppna textfilen .xinitrc från hemkatalogen, eller skapa den om det behövs. Lägg till raden:

```
exec ${KDEDIR}/bin/startkde
```

VIKTIGT

Det kan vara nödvändigt att utföra samma steg med filen .xsession, också i hemkatalogen. Det är särskilt sant om grafiska inloggningshanterare som sddm, gdm eller xdm används.

Starta nu det nya KDE: i BSD- och Linux®-system med stöd för virtuella terminaler, används tangentkombinationerna **Ctrl**+**Alt**+**F1** ... **Ctrl**+**Alt**+**F12** för att byta till virtuell konsoll 1 till och med 12. Det låter dig köra fler än en skrivbordsmiljö samtidigt. De första sex är textterminaler, och de följande sex är grafiska skärmar.

Om den grafiska inloggningshanteraren visas istället när du startar datorn, kan du använda den nya KDE-miljön även om den inte anges som ett alternativ. De flesta inloggningshanterare, inklusive sddm, har ett alternativ att använda en 'Egen session' när du loggar in. Med det alternativet

laddas dina sessionsinställningar från filen `.xsession` i din hemkatalog. Om du redan har ändrat filen som beskrivs ovan, ska alternativet starta den nya KDE-installationen.

Om det inte gör det, finns det något annat som du kan prova som normalt ska fungera: Tryck på **Ctrl**+**Alt**+**F2**, så visas en textterminal. Logga in som den särskilda användaren och skriv:

```
startx -- :1
```

TIPS

Du kan köra KDE från källkod och det gamla KDE samtidigt. Logga in som din vanliga användare, och starta det stabila KDE-skrivbordet. Tryck på **Ctrl**+**Alt**+**F2** (eller **F1**, **F3**, etc.) så visas en textterminal. Logga in som den särskilda KDE Git-användaren och skriv:

```
startx -- :1
```

Du kan gå tillbaka till KDE-skrivbordet för din vanliga användare genom att trycka på snabbtangenten för skrivbordet som redan kör. Den är normalt **Ctrl**+**Alt**+**F7**, men du kanske måste använda **F6** eller **F8** istället. För att returnera till KDE kompilerat med kdesrc-build, ska du använda samma sekvens, utom med nästa funktionstangent. Om du till exempel behövde skriva **Ctrl**+**Alt**+**F7** för att byta till det vanliga KDE, skulle du behöva skriva **Ctrl**+**Alt**+**F8** för att gå tillbaka till KDE byggt med kdesrc-build.