

Inhoud

[Intro 3](#_Toc179538815)

[MXE compiler Ubuntu 24 3](#_Toc179538816)

[MXE compiler Ubuntu 24 gefaalt 6](#_Toc179538817)

[MXE compiler Ubuntu 22 8](#_Toc179538818)

[Mxe uitleg 9](#_Toc179538819)

[Make uitleg 12](#_Toc179538820)

[Test case 14](#_Toc179538821)

[Conclusie 20](#_Toc179538822)

[Link theorie 22](#_Toc179538823)

# Intro

Ik ga een to do list app maken. Hierbij kan je een taak toe voegen, verwijderen of aanduiden dat deze gemaakt is. Naast jouw eigen taken kan je ook een smiley zien welke aangeeft hoe het met je progressie van de te taken die je in het totaal moest doen.

Zo geeft de smiley 3 stemmingen weer:

1. Hij is blij en groen als je meer dan 60% van je taken al gedaan hebt.
2. Hij is neutraal en oranje als je tussen de 40% en de 60% zit.
3. Hij is sip en rood als je minder als 40% van je te doende taken maar gedaan hebt.

Ook kan je de taken in een file gaan wegschrijven. Zo kan je deze altijd terug kijken. Of opnieuw gaan inlezen.

# MXE compiler Ubuntu 24

1. Ik heb eerst een ubuntu machine gemaakt van versie 24.04.1 LTS
2. Vervolgens heb ik de nodige packages geinstaleerd.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

2.1) Na het krijgen van de error dat python3-disutilities niet gevonden kan worden heb ik deze vervangen door python3-setuptools. Waarna alles is kunnen downloaden.

1. Als we alle packeges hebben geïnstalleerd dan kunnen we de mxe repository gaan halen van github.



1. Als dit gelukt is gaan we in de map van mxe gaan die onder deze directory is in dit geval de home directory van de huidige user. Als we in deze map zitten gaan we make -j6 qt6 gebruiken.



Hierbij staat de make voor het maken van de nodige make files Hierbij staat de make voor het laten instaleren van alle nodige packages voor later make te kunnen gebruiken. Waardoor dat je dus uiteindelijk alle cpp, header en ui files kan combineren naar object files en uiteindelijk ook naar 1 exe bestand. Nu wordt nog niets alle tot 1 exe gemaakt maar alles wat er voor nodig is. Wat dus de build-tool is.

De optie -j6 is om te zeggen dat je de build van make voor de nodige files kan doen met 6 cores. Waardoor dit sneller kan gedaan worden.

Parameter qt6 is om te zeggen dat je niet alles moet builden van make maar enkel het gedeelte dat nodig is voor qt te kunnen builden en specifiek qt versie 6.

1. Vervolgens heb ik ervoor gezorgd dat ik mijn cpp, header en ui files op de ubuntu vm heb kunnen krijgen. Als dit gelukt is ben ik in de commandline naar deze locatie gegaan waar dat alle files staan. En doe je dit export commando.



Dit is om ervoor te zorgen dat je in deze shell duidelijk maakt waar mxe en alle nodige bestanden staan. En zal dus eerst zoeken in deze map of een child map of je een script vindt om te kunnen uitvoeren. Je moet dit commando elke keer (als je een nieuwe shell opent) uitvoeren. Je zou dit ook in een script kunnen zetten zodat je dit elke keer moet uitvoeren maar dit automatisch gedaan kan worden. $PATH achteraan is om er voor te zorgen dat de locatie van de andere / standaard uitvoerbare bestanden er terug wordt achter geplakt zodat je verder nog alles kan doen.



Hierdoor maak je duidelijk aan het systeem dat je qmake gebruikt van de mxe toolchain. Voor de make file te gebruiken.

1. Make uitvoeren. Om alles tot 1 exe te krijgen.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

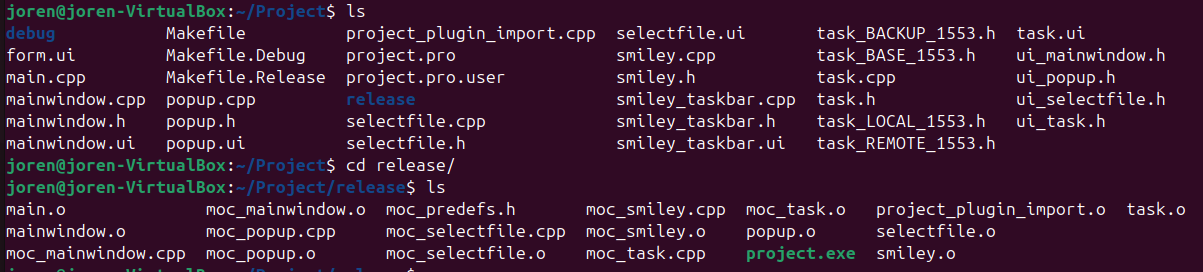
Als je make in de map met alle bestanden uitvoerd gaat hij deze tot 1 exe gaan laten vormen.

Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving

Als je nog eens make doet dan gaat hij zeggen dat alles direct gedaan is. Wat het ook makkelijk maakt dat als je maar 1 cpp file hebt aangepast dat dit niet alles opnieuw moet builden maar weet dat enkel deze cpp file veranderd is en dus ook simpel en snel deze file moet gaan builden. Om in de object file te zetten.

1. Exe vinden.



Als we gaan zoeken waar nu de exe staat kunnen we deze vinden in de release map.

# MXE compiler Ubuntu 24 gefaalt

Ik heb eerst op een ubuntu de mxe toolchain proberen te gebruiken maar dit lukte niet van wegen een qtbase error. Voor zowel qt(5) als qt6. Maar na het volledig opniew maken van mijn ubuntu vm is het dus wel gelukt. Zoals de mxe compiler ubuntu 24 hierboven aangeeft.

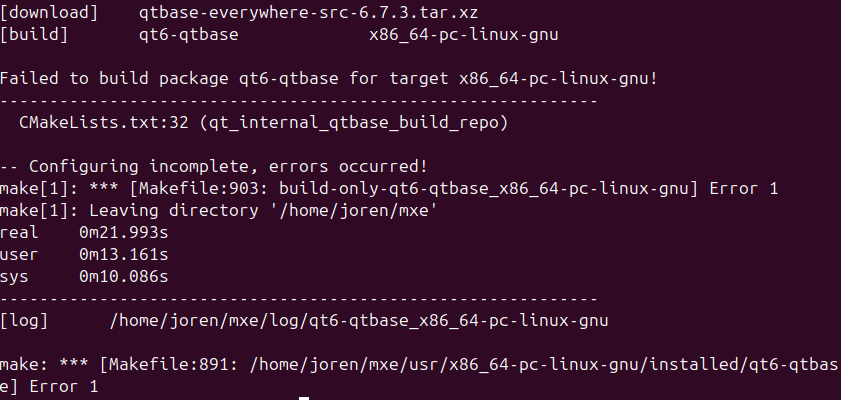
1. Instaleren van de nodige requirements.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Hierbij kreeg ik ook de error van python-disutilities. Maar heb ik deze gewoon weggedaan.

1. Git clone. Hierbij heb ik ook de git clone die werkte.
2. Make -j6 qt(6)

Na he proberen van het make commando voor zowel qt5 als qt6 te proberen uitvoeren bleef ik de error krijgen dat qtbase niet gebuild kan worden.  
  


Hierbij heb ik geprobeert alles nog een weg te doen en opnieuw te instaleren maar dit is niet gelukt.

# MXE compiler Ubuntu 22

Na dat het gefaald was op ubuntu 24 ben ik op internet gaan zoeken naar een andere oplossing en heb ik een manier gevonden voor versie 22 van ubuntu.

1. Requirements

Zoals bij de vorige mxe compilers uitleggen heb ik de nodige requirements die ze op de website hebben gezet gedownload. Maar hierbij op versie 22 van ubuntu heb ik geen error gekregen dat python3-disutilities niet gevonden kon worden.

1. Git clone

De mxe toolchain ook weer hierbij van de github gekloond op mijn vm.

1. Git reset

Na ik in de mxe map ben gegaan heb ik nog het onderstaande commando ingegeven.



Door dit commande wordt mxe naar een de commit dbe2413 waardoor dat de mxe uiteindelijk wel gelukt is.

1. Make

Zoals bij de ubuntu versie 24 waarbij het wel gelukt is moet je hier dus ook verder de make -j6 qt6 doen en al het verdere tot op het einde.

# Mxe uitleg

Onder de mxe folder ben ik eens gaan rond kijken wat erin staat. Zo heb ik gevonden waar alle binutils staan. Deze staan onder de map usr/bin in mxe. De binutils zijn eigenlijk alle programma’s die nodig zijn om jouw qt files te kunnen linken en compilen bij één gezet.

Afbeelding met tekst, schermopname, menu, Lettertype

Automatisch gegenereerde beschrijving

Hierna ben ik gaan proberen zoeken hoe mijn exe is opgebouwd geweest. Ik heb gevonden om info te krijen over hoe het gebouwd is door middel van readelf. Dit werkte niet omdat readelf voor linux exe bestanden wilt inlezen en dit gaan linux bestand is. Zo heb ik verder gezocht en heb ik gevonden dat objdump wel werkt. Zoals je kan zien in de afbeelding hieronder.

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving

Hier heb ik gekeken uit welke sectie headers de exe bestaat. In de sectie text staat de uitvoerbare code, in data staan de variabelen en globale data. Rdata zijn de constante variabelen, bss zijn de variabelen die geen initiele waarde hebben gekregen, edata zijn de functies die vanuit andere pragramma’s worden aangeroepen. Idata idata is voor de functies die uit andere libraries, reloc is voor het aanpassen van de symbolen afhankelijk van locatie waar de exe wordt uitgevoerd.

De naam van de toolchain die we gebruiken bij onze mxe cross-compile is i686-w64-mingw32.static dat kan je zien tijdens het uitvoeren van het make qt6 commando. Waarbij er op het einde staat welke toolchain je gebruikt.



Je kan hieruit opmaken dat je een 32bit windows zowel een 64 bit windows omgeving of een 32 bit windows gaat naar compilen afhankelijk van de architecure. En static wat er achter staat wilt zeggen dat je de nodige libraries direct mee in de exe gaat zetten.

Ja kan meer info vinden over de tool chain als je het commando in de afbeelding hieronder uitvoert. Hierbij is de optie -v voor verbose oftewel een gedetailleerde info.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Hier zou iets van een glibc moeten staan.

Daarnet hebben we in de mxe/usr de binutilities gevonden onder /bin. Maar als we onder /lib gaan kijken zien we alle c libraries die we gebruiken.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Om dit duidelijker te kunnen weer geven heb ik enkel de files met .a (de statische libraries) of .so (de dynamische libraries) als extensie weergegeven.

Afbeelding met tekst, schermopname, Lettertype

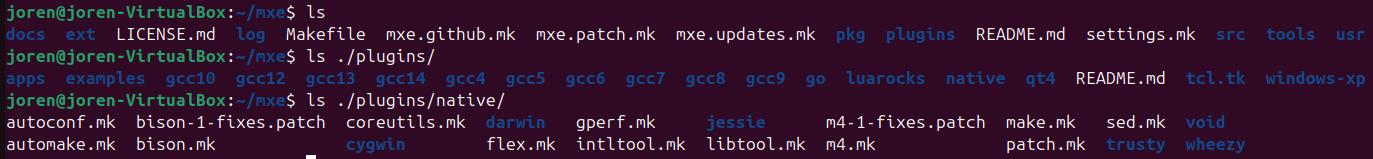
Automatisch gegenereerde beschrijving

Een andere belangrijke librarie is de glibc, omdat deze librarie gebruikt wordt op linux door vrijwel alle c en c++ programma’s (zoals bestanden beheer, geheugenbeheer …) dewelke je kan door ldd –version te doen of locate libc.so en locate.a

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Ldd --version gaat controleren of de glibc librarie bestaat en welke verie deze heeft. Terwijl met de locate functie kan je alle bestanden zijn van libc.so en .a de reden dat het enkel libc noemt en niet glibc is omdat de libc bestanden gebruikt worden om de glibc functies op te roepen.

Als we in de mxe folder nog wat gaan voort zoeken kunnen we in de plugins van mxe een folder vinden met native. En hierin kunnen we zien dat mxe ook gebruikt maakt van autotools.  


Dit zit hier omdat het een makkelijke manier is voor portabiliteit (aldus het cross compileren). Sommige libraries niet beschikbaar kunnen zijn overal of simpelweg niet bestaan. Ook kan je de 3 herkenbare systeem types (build, host, target) instellen waardoor het dus ook weer makkelijker maakt om te kunnen cross compilen. Dit is handig omdat mxe wilt compilen op een linux om te kunnen gebruiken op een windows.

Autoconf.mk is het bestand/tool die er voor zorgt dat er autommatisch een configure script genereert voor je project.

Automake.mk is het bestand/tool die automatisch Makefile.in bestanden genereert op basis van de sjablonen die zijn gedefinieerd in de Makefile.am bestanden. Hierdoor wordt een Makefile bestand makkelijker gemaakt.

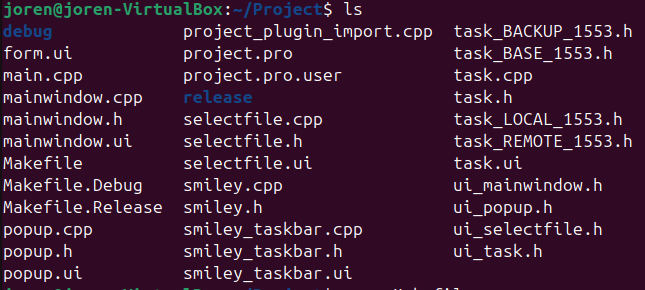
Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, Lettertype

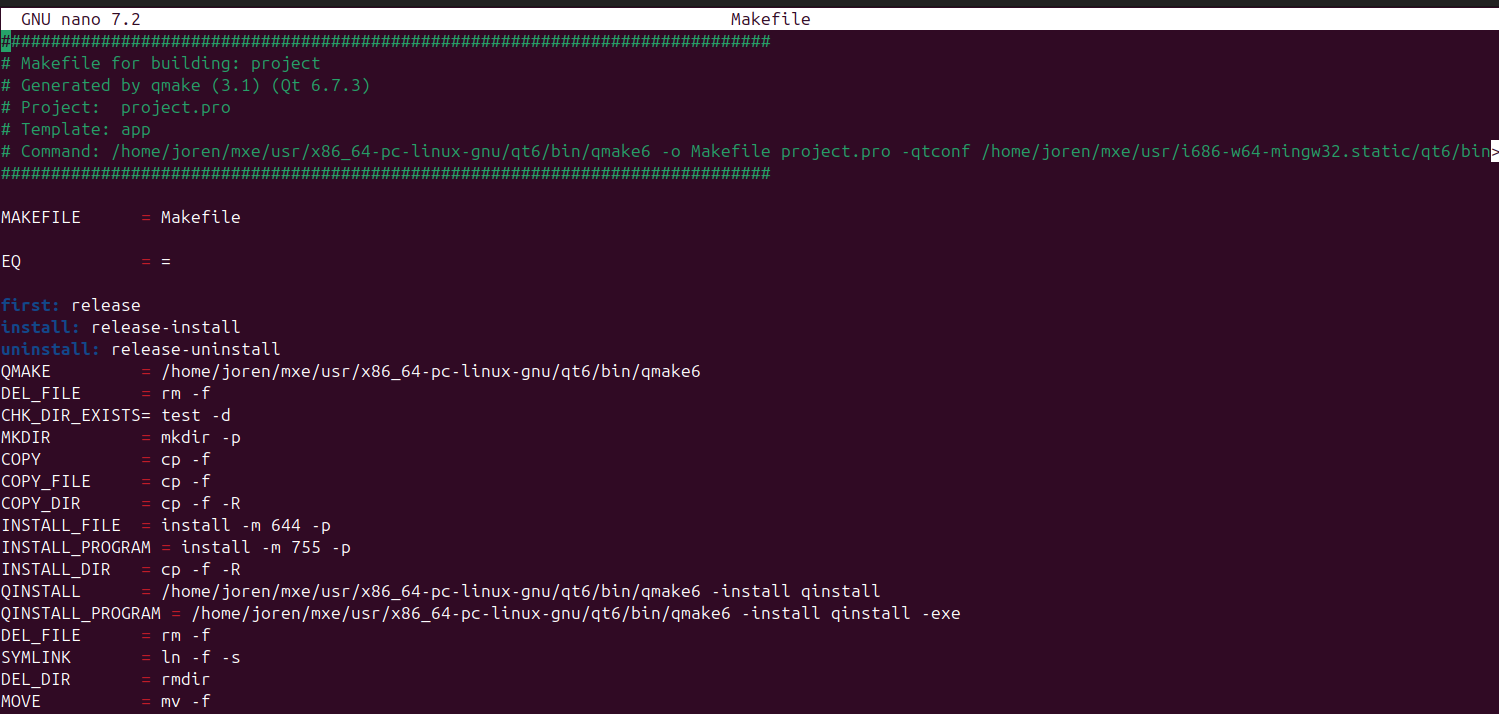
Automatisch gegenereerde beschrijving

# Make uitleg

Als we in de project folder gaan kijken waar we de make gedaan hebben kunnen we de Makefile zien. Dit zijn er 3.



Als je gewoon make uitvoert dan wordt de Makefile opgeroepen. We kunnen eens in de make kijken hoe de exe gemaakt wordt.



In de Makefile zou ik zelf niets direct kunnen vinden hoe deze wordt gemaakt. Maar de “first: release” is wel belangrijk, dit geeft aan dat als eerste (als we geen argumenten meegeven) we de release file moeten gaan aanroepen. En deze staat dan onder Makefile.release

Afbeelding met tekst, schermopname, software

Automatisch gegenereerde beschrijving

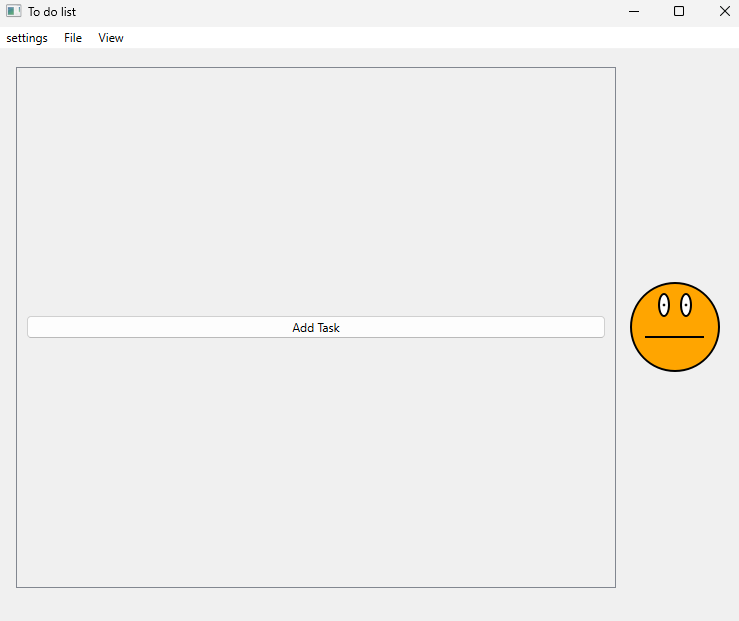
Afbeelding met tekst, schermopname, Lettertype, menu

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

In deze release Makefile vinden we info over de compiler. Deze staat onder de CC en de opties van de compiler dewelke onder CFLAGS staan. Maar ook de source files de header files die gebruikt worden, de object files. En ook de naam executefile kunnen we hier zien.

# Test case

Ik heb de app gestart. En gekeken of de smiley standaard neutraal is.

Vervolgens wil ik een taak toevoegen. Dit kan ik doen door middel van op de add knop te duwen. En kan ik een titel en een beschrijving van mijn nieuwe taak toevoegen.

Afbeelding met tekst, schermopname, scherm, software

Automatisch gegenereerde beschrijving

Als ik op add klik wordt de taak toegevoegd aan de lijst om alle taken te zien en wordt de smiley rood en sip.

Afbeelding met tekst, schermopname, scherm, software

Automatisch gegenereerde beschrijving

Ik kan ook een nieuwe taak toevoegen via het menu bovenaan het scherm door de view te veranderen of de shortcut. En de titel en de beschrijving zou terug leeg moeten zijn.  
  
Afbeelding met schermopname, tekst, lijn

Automatisch gegenereerde beschrijving

Als ik een lege taak probeer in te voegen zonder een titel zou ik een error moeten krijgen dat je een titel moet hebben. Je kan deze error niet weg klikken tot je op ok geklikt hebt.

Afbeelding met tekst, schermopname, scherm, software

Automatisch gegenereerde beschrijving

Je kan terug naar het home / taak overzicht scherm gaan door middel van op de back knop te duwen onderaan de pagina. Bovenaan de pagina in het menu onder view zou je ook terig kunnen gaan door middel van op home te klikken of de shortcut in te voeren.

Afbeelding met tekst, schermopname, scherm, software

Automatisch gegenereerde beschrijving

Als ik op een klaar check box van een taak klik dan zou het mogelijk kunnen zijn dat de smiley verandert. Dit is natuurlijk afhankelijk van het % van de voltooide taken.

Afbeelding met tekst, schermopname, software, scherm

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, software, Computerpictogram

Automatisch gegenereerde beschrijving

Als ik op delete duw van een taak dan zou deze taak weg moeten zijn en het mogelijk de smiley aan passen.

Afbeelding met tekst, schermopname, software, scherm

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, scherm, ontwerp

Automatisch gegenereerde beschrijving

Je kan ook het scherm veranderen van groote met een minimum grote van 1 taak.

Afbeelding met tekst, schermopname, scherm, software

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, software, Computerpictogram

Automatisch gegenereerde beschrijving

Ik heb ook getest dat ik het kan opslaan in een file door middel van op de file menu opslaan te selecteren. Of de bij horende shortcut. Als ik opslaan kies dan gaat de to do lijst worden opgeslagen in de file die geslecteerd is (van eerder geopend of al reeds opgeslagen file of nieuwgekozen file). Als je opslaan als klikt dan moet je sowieso een nieuwe file selecteren om in op te slaan. Hierbij zijn de popup schermen modaal. Zodat je er niet uit kan zonder dezen te bevestigen of weg te kliken.

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

Als je nieuw een file gaat opslaan (of opslaan als klikt) krijg je een popup om een file te selecteren. Om op te slaan. Je kan zelf een file noteren. Of op de 3 puntjes kliken en dan gaan zoeken/ selecteren via een ui om een txt bestand op te slaan. Als je via de 3 puntjes een locatie selecteerd dan wordt deze ook in de balk getoond.

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Als je op save klikt dan wordt het bestand opgeslaan waar jij hebt gezegd. En ziet het er als voligt uit.  
Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

Als je op cancel klikt krijg je een extra vraag of je zeker bent om het bestan niet op te slaan. Waar bij je op ok klikt en het dus niet wilt opslaan of op cancel klikt en het niet gaat opslaan.

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

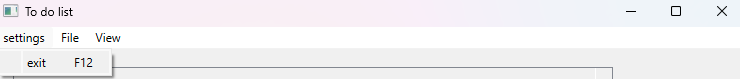
Als je op het file menu klikt en file open word je direct gevraagd om een file te selecteren van het type txt. Deze file moet in de volgende vorm staan zodat hij weet welk de titel, taak is. En deze nieuwe taken worden dan in de lijst van de app toegevoegd.

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

Tot slot heb ik gekeken dat de app op een juiste manier wordt afgesloten door middel van op het kruisje te duwen recht boven, de shortcut f12 of de exit knop in de settings menu te duwen.



# Conclusie

Hoewel de app hetzelfde doet en dezelfde functie’s heeft zijn er toch kleine verschillen.

|  |  |  |
| --- | --- | --- |
|  | Windows | Linux mxe compiled |
| Exe grote |  |  |
| menubar |  |  |
| Exe uitvoeren zonder qt te starten | Afbeelding met tekst, schermopname, Lettertype  Automatisch gegenereerde beschrijving |  |
|  |  |  |

De reden van deze kleine verschillen is omdat de exe voor windows native wordt gebuild, de exe van mxe cross gebuild wordt.

Één van de reden in het verschil van de exe grote is omdat windows native gebuild wordt, voor mxe wordt cross build gebruikt.

Een andere reden is als je de

Toe voegen verschil grote:  
Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

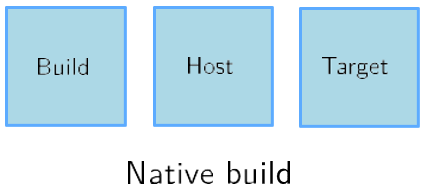
Dit wilt zeggen dat je de Qt6Widgets.dll niet kan vinden. Dll staat voor dynamic link library wat wilt zeggen dat deze code gedeeld wordt tussen verschillende projecten.

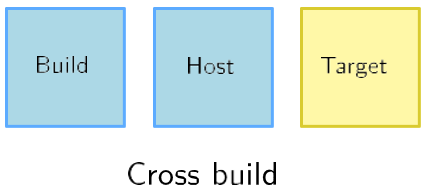
Dit is dan ook een kern van het probleem. De Qt6Widgets.dll file weet windows niet waar hij deze kan vinden. Deze staan niet in de folder van de exe nog in de path variabelen. Je zou dit dan ook zo kunnen oplossen door de path variabelen aan te passen dat windows weet dat hij in de qt folder ook moet kijken of door de nodige dll files bij de exe te plaatsen.

Dus concreet is de grootste reden van het verschill in grote tussen de exe is dat de windows exe dynamic link library gebruikt waardoor de exe kleiner is en cross compile exe groter is omdat deze static link library gebruikt en dus de libraries in de exe toevoegd.

Als het niet gelinkt is aan de path variabelen kan hij dit niet vinden en wordt dit gecrashed.

Een ander deel van de reden is omdat zoals eerder gezegd de windows exe dynamic compiled wordt en dus tijdens het uitvoeren kan de code die gegenereerd is door de compiler veranderen ( naar de dll). Voor de cross compile exe staat er bij de compile .static achter, dit wilt zeggen dat de code die de compiler genereerd niet tijdens het uitvoeren van de exe verandert kan worden.

Native build houdt in dat je zowel het builden als uitvoeren als compilen voor jouw systeem gemaakt wordt. Hierdoor is de exe meer geoptimaliseerd voor het systeem en dus ook de exe kleiner is. Maar ook omdat er al frameworks op het systeem reeds geinstaleerd is en deze dus niet meer moet. Het nadeel van deze build methode is dat je de exe enkel op jouw buildsysteem kan uitgevoerd worden en niet door iets anders.

Cross build houdt in dat je de app gaat builden en compilen buiten jouw systeem (linux). En uitgevoerd wordt op jouw systeem (windows). Het voordeel van dit soort build methode te gebruiken is dat je het buiten jouw build syssteem kan gebruiken, je kan dus ook software ontwikkelen voor een ander platform. Het nadeel van deze build methode is dat je zelf de juiste toolchain (mxe) moet vinden voor het buildsysteem. De exe kan groter zijn omdat je de nodige libraries mee in de exe moet zetten. De libraries die gebruikt hebt tijdens de build van qt zelf (native build) kan ook niet altijd door de cross build (mxe) gebruikt worden. Hiervan heb ik geen last gehad.

# Link theorie

Dit is uit gelegd in vorige punten.

Blauw: nieuw geschreven

Geel: verbeteringen van de leerkracht

Rood: mag weg