Erno de Weerd is trainer en consultant bij Info Support.

Services For UNIX 3.5 voor ontwikkelaars

UNIX-CODE DRAAIEN OP HET WINDOWSPLATFORM

Dit artikel beschrijft welke functionaliteit Services For UNIX 3.5 (SFU) beschikbaar stelt en hoe deze services gebruikt kunnen worden bij de migratie van Unix-programma's. Elders in dit blad staat het artikel van Tom Werner over de architectuur-, installatie- en beheeraspecten van SFU 3.5. Daarin komt alles aan de orde over de structuur van SFU 3.5. In dit artikel richten we ons op de development-kant van Services For UNIX.

Op het eerste oog is, na installatie van SFU, een eenvoudige Unix-omgeving beschikbaar gekomen. Om dit te demonstreren gaan we eerst, volgens goed gebruik, kijken naar een 'Hello World'-variant.

Shell

Na volledige installatie van SFU 3.5 vinden we in het startmenu onder programma's bij Windows Services for UNIX een tweetal shells: de *C Shell* en de *Korn Shell*. Voor de voorbeelden in dit artikel maakt het niet uit welke gekozen wordt. Er zijn drie C/C++-compilers beschikbaar: cc, c89 en gcc (g++). De eerste twee maken gebruik van de C++-compiler van Microsoft uit Visual Studio. Om van cc en c89 gebruik te kunnen maken, moet Microsoft Visual C++ dus geïnstalleerd zijn. In de voorbeelden in dit artikel gaan we uit van de gcc-compiler met de waarschuwing dat de gcc-compiler onder de 'Free Software Foundation's General Public License' valt. De precieze consequenties hiervan zijn te lezen in de docs-folder (gpl.txt).

Om in de Shell gebruik te kunnen maken van het Windows filesysteem vinden we in de directory /dev/fc alle schijfeenheden zoals we die onder Windows kennen. Zo is bijvoorbeeld: c:\windows te vinden als /dev/fc/C/WINDOWS. Pas hierbij op dat Unix hoofdlettergevoelig is! De linkers van Unix maken onderscheid tussen de bestanden hello.c en Hello.c, de linker van Visual Studio kan dat niet. De enige oplossing in zulke gevallen is het verplaatsen of hernoemen van de bestanden. In het algemeen is het ook aan te raden zo min mogelijk gebruik te maken van paden met spaties; niet alle Unix-commando's kunnen daarmee overweg. Soms is het gebruik van dubbele aanhalingstekens ("/dev/fs/C/Program Files") of het gebruik van 8.3-bestandsnamen (/dev/fs/C/PROGRA~1) daar een oplossing voor. Bij het configureren van environment-variabelen is het gebruik van 8.3 namelijk de enige oplossing.

```
#include <stdio.h>

int main()
{
  printf("Hello UNIX World\n");
  return 0;
}
```

Codevoorbeeld 1.

Hello UNIX!

De code uit codevoorbeeld 1 is een klassieke "Hello World"-variant. Nadat we het bestand hebben gemaakt met een tekstverwerker zoals vi, Notepad of Visual Studio en hebben bewaard als /dev/fs/C/unixtest/hello.c, kan de applicatie gecompileerd worden met:

```
gcc -Wall -o /dev/fs/C/unixtest/hello /dev/fs/C/unixtest/hello.c
```

De –Wall-optie zorgt ervoor dat alle waarschuwingen van de compiler afgebeeld worden. In dit geval zou er geen enkele foutmelding mogen optreden. De –o-optie vertelt hoe de outputfile moet heten en de laatste parameter is de inputfile. De applicatie kan vervolgens gestart worden met: /dev/fs/C/unixtest/hello Wanneer alle bestanden in de huidige directory staan kan het compileren op de volgende manier plaatsvinden:

```
gcc -Wall -o hello hello.c
```

Het starten van de applicatie gaat dan zo:

```
./hello
```

Een stap verder

In de "Hello World"-applicatie maken we geen gebruikt van Unix-specifieke zaken. Om te laten zien dat SFU 3.5 een echte Unix-omgeving biedt, gaan we daarom in het volgende voorbeeld een stap verder.

De functie *fork()* start een nieuw kind-proces. Een dergelijk kind-proces kennen we niet onder Windows. Onder Windows zijn processen niet gerelateerd. De functie *fork()* geeft een waarde van het type pid_t terug en deze variabele bevat het id van het gestarte proces. Onder Unix is het kind-proces een (bijna) exacte kopie van het ouder-proces. Het kind-proces start dan ook precies daar waar de *ouder* gebleven was; direct na de *fork()*. Eén van de verschillen is dat de teruggegeven id in het kind-proces 0 is en dat hij in het ouder-proces of negatief is (fout) of positief (id van de ouder). Met behulp van deze id is dus te bepalen of we in het ouder-proces zijn of in het kind-proces.

In codevoorbeeld 2 wordt er twee keer *fork()* gedaan. Dit is om een volledig onafhankelijk proces (kleinkind-proces) te maken. Op deze manier start onder Unix bijvoorbeeld een zogenaamde dae-

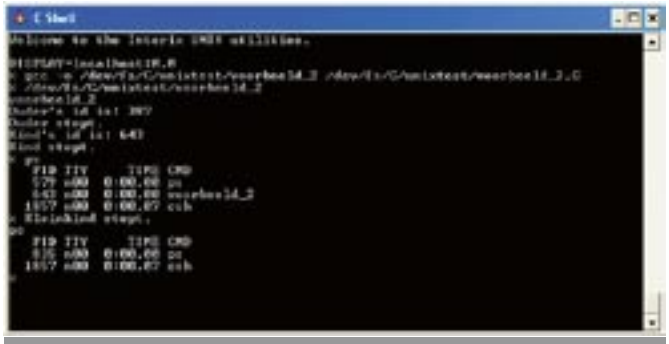
mon. Een daemon is vergelijkbaar met een Windows Service. Het compileren gaat op dezelfde manier als bij codevoorbeeld 1. Na het starten van de applicatie zal pas na 10 seconden het kleinkind-proces zich afmelden en eindigen. Door binnen die 10 seconden het commando *ps* te geven zien we in het overzicht van de lopende processen 'ons' kleinkind-proces.

Migratie

Het ligt niet voor de hand dat er met SFU 3.5 voortaan voor Unix geprogrammeerd gaat worden op het Windows-platform. De functionaliteit die we in de bovenstaande voorbeelden zien is bedoeld om code, geschreven voor Unix, zonder al te veel moeite op een Windows-machine te kunnen laten draaien. SFU is wat dat betreft een tussenstation bij migratie van de Unix-wereld naar de Windows-wereld. In het ideale geval is het compileren van de Unix-broncode en het juist installeren (configuratiebestanden bijwerken) van de applicatie voldoende om de applicatie te laten draaien op Windows. Vaak echter zullen de make-files (configuratiebestanden voor de compiler) moeten worden aangepast. De make van

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

...

void kind()
{
   pid_t pid = fork();
   if(pid < 0)
   {
      fork_fout(pid);
   }
   else if(pid != 0)
   {
      printf("Kind's id is: ");
      printf("%d\n", (int)pid);
      printf("Kind stopt.\n");
      _exit(0);
   }
   else
   {
      sleep(10);
      printf("Kleinkind stopt.\n");
      _exit(0);
   }
}

...

int main()
{
   pid_t ppid = fork ();
   if(ppid < 0)
   {
      fork_fout(ppid);
   }
   else if(ppid == 0)
   {
      kind();
   }
   else
   {
      ouder(ppid);
   }
   return 0;
}
```

Codevoorbeeld 2.



Afbeelding 1. Kleinkind stopt

de Interix SDK voldoet aan de Berkeley Software Distribution (BSD) 4.4-versie en voldoet ook aan de Posix.2-standaard. Als de oorspronkelijke make-files niet voor deze standaarden gemaakt zijn, kan het dus betekenen dat de make-files aangepast moeten worden. De aanpassingen die nodig kunnen zijn staan in de documentatie van SFU 3.5.

Ook kan zich de situatie voordoen dat broncode aangepast moet worden. In zulke gevallen kunnen we gebruikmaken van de macro __INTERIX, die standaard aanstaat voor cc, c89 en gcc. Door te testen op deze macro (#ifdef __INTERIX) kan er in de broncode op een nette manier een aanpassing gemaakt worden.
In de documentatie van SFU 3.5 staat een (korte) lijst van specifieke aanpassingen die wellicht nodig zijn om de Unix-code te kunnen compileren met behulp van de Interix SDK. Vooral daar waar Unix en Windows elkaar ontmoeten - denk aan beveiliging, bestandssysteem, poorten en dergelijke - zullen waarschijnlijk aanpassingen gemaakt moeten worden.

Een opvallende afwezige in SFU 3.5 is een X server. Microsoft heeft ervoor gekozen om geen X-server mee te leveren om de volgende drie redenen:
1. Een commerciële versie zou SFU 3.5 duurder maken terwijl niet iedereen behoefte heeft aan een X-server.
2. Een freeware X server zou Microsoft verplichten tot support van de server als onderdeel van SFU 3.5 waardoor de prijs ook zou stijgen.
3. Het door Microsoft zelf bouwen en onderhouden van een X server is ook erg kostbaar.

Een oplossing voor dit probleem ligt bij de gebruiker van SFU 3.5. Afhankelijk van de specifieke eisen (snelheid, drivers, support, OpenGL-ondersteuning) kan deze kiezen voor een commerciële X server of een freeware X server. Voorbeelden van commerciële X servers zijn Exeed van Interop Systems en X-Win32 van Starnet. De bekendste freeware X server is XFree86.
Verder is er bij Interix een grote verzameling applicaties gratis beschikbaar die geschikt zijn voor SFU 3.5. In deze verzameling zit bijvoorbeeld de broncode van de bekende webserver Apache, klaar om te draaien op SFU.

In het tweede voorbeeld zagen we hoe een Unix-daemon gemaakt kan worden. Bij het migreren van een daemon staan we voor de volgende keus: gaat de daemon draaien als een daemon op SFU 3.5 of willen we de daemon 'upgraden' tot een Windows Service? Beide zijn mogelijk. Het voordeel van de eerste optie is dat

Gratis versie van Windows Services for UNIX 3.5
Bestel een gratis exemplaar van de complete versie van Services for UNIX 3.5. SFU bevat alle tools, compilers, en cross-platform netwerk-services voor het integreren van Windows- en Unix-omgevingen.

U kunt SFU 3.5 downloaden of bestellen op: http://www.microsoft.com/netherlands/windows/sfu/download.aspx
De cd-rom met Services for UNIX 3.5 wordt u kosteloos aangeboden door Microsoft Nederland.

de broncode niet of nauwelijks aangepast hoeft te worden. Het nadeel is dat de daemon niet draait onder een Windows-account waardoor de daemon niet eenvoudig gebruik kan maken van zijn Windows-rechten voor het benaderen van resources op het netwerk. Kiezen we voor de migratieoptie dan zal de service als een echte Windows Service draaien en dus ook een Windows-account hebben, maar mag de service onder meer *fork()* en *exit()* niet meer aanroepen. Dit heeft tot gevolg dat de Unix-gebruiker de broncode moet aanpassen.

Samenvatting

SFU 3.5 biedt op een bijzonder fraaie manier de mogelijkheid om code, geschreven voor Unix, te migreren naar het Windows-platform.

Erno de Weerd

MSCD (Visual Studio 6 en Visual Studio .NET) en MCT is trainer en consultant bij Info Support (www.infosupport.com). Zijn e-mailadres is ernow@infosupport.com

Nuttige internetadressen
Services For UNIX 3.5: www.microsoft.com/windows/sfu
Overzicht van nieuwe functies: www.microsoft.com/netherlands/windows/sfu/nieuw.aspx
Reviewers guide: http://www.microsoft.com/windows/sfu/techinfo/revguide.asp
Application migration: http://msdn.microsoft.com/library/en-us/dnucmg/html/ucmglp.asp
GPL: www.gnu.org/licenses/gpl.html
GCC: www.network-theory.co.uk/docs/gccintro/index.html
UNIX programming FAQ: www.erlenstar.demon.co.uk/unix/faq_toc.html
Interix Interop Toolworks: www.interix.com/InteropToolworks.htm