Processer

under Microsoft Windows XP™

Henrik Bäck
850611-6253

Inlämningsuppgift i**:**
Operating Systems; DAVB01
vid **Karlstads Universitet**, VT 2005

Korrekturläsning: **Mathias Andersson**
2005-04-12

Handledare: **Thijs Holleboom**
Nils Dåverhög
Hans Hedbom

Inlämnad **2005-04-14**

Antal ord i denna uppsats: **807** st

Innehållsförteckning

Processer under Windows XP

Det som under Windows XP räknas som en process är ett program som för tillfället körs och har en tråd av kod som kan schemaläggas av operativsystemet. Varje process har ett virtuellt minnesutrymme och information för att initiera alla trådar som kan komma att köras av programmet. För varje process finns en eller flera trådar som kan körs.

Det finns flera anrop som kan göras via Application programming interface(vidare kallat API) för att hantera processer. API är en samling definitioner som kan användas för att göra systemanrop. Olika system med samma uppsättning API kan använda samma anrop för att utföra de egentliga systemanropen som skiljer sig åt från system till system.

Processhanteraren

Processhanteraren tillhandahåller tjänster för att skapa, använda och ta bort processer. Den har ingen kunskap om hur processer hänger samman eller hur schemaläggning av trådar sköts, däremot sätter den prioritet för processer. Schemaläggningen av trådar sköts av kärnan.

Processhanteraren köar och tillhandahåller tjänster för att initiera trådkörning, slutföra I/O, avsluta trådar och processer. Den kan också se till att ansluta en *debugger* till en process, denna kan pausa och återuppta kodexekvering av trådar samt skapa trådar som befinner sig i ett viloläge.

Skapa processer

För att en process skall kunna startas måste en annan process anropa systemanropet `CreateProcess()`. Detta anrop skapar en ny process med en ny primär tråd, en egen stack för tråden samt laddar in de bibliotek som används av processen. Storleken på stacken är som standard 1 megabyte men storleken kan varieras genom att ange det som ett argument till `CreateProcess()`.

Det finns flera olika prioriteringsklasser som en process kan köras i. Alla dessa klasser tillhandahålls dock inte när en process skall skapas via API. När man skapar en process som ett barn till en huvudprocess kan man välja mellan IDLE_PRIORITY_CLASS, NORMAL_PRORITY_CLASS, HIGH_PRIORITY_CLASS eller REALTIME_PRIORITY_CLASS. Dotterprocessen som skapats kommer hamna i samma prioriteringsklass som moderprocessen. Vanligtvis ligger processerna i NORMAL_PRIORITY_CLASS, dock kan dotterprocessens initieringsprioritet påverkas genom att ange detta i anropet.

Det går bra att ändra prioriteringen på dotterprocessen i efterhand med hjälp av `SetPriorityClasss()`-funktionen. Dock kan endast priviligerade användare och administratörer ta en process in i REALTIME_PRIORITY_CLASS.

Det är även möjligt att ange vilken prioritetsklass en process skall startas igenom att starta processen med hjälp av kommandot `START`.

Schemaläggning av processer

Eftersom en process som körs av en användare oftast är ett interaktivt program måste systemet kunna ge denna process tillräckligt med resurser för att köras smärtfritt. En process i klassen NORMAL_PRIORITY_CLASS som körs i förgrunden och är aktiv på skärmen får därför mer tid att köras än andra processer innan den byts ut och köas. Oftast får en förgrundsprocess tre gånger längre tid i processorn än en bakgrundsprocess i samma prioriteringsklass. Under Windows XP finns dock en inställning där det är möjligt att ange om Windows skall optimeras för processer i förgrund eller bakgrund.

Processens trådar kan ha sex olika lägen. Dessa lägen är som följer:
Redo (Ready)
Redo indikerar att en tråd är redo att köras när den får tillfälle.
Beredd (Standby)
Beredd indikerar att en tråd står på tur att köras härnäst
Körs (Running)
Körs indikerar att en tråd exekverar för tillfället.
Väntande (Waiting)
Väntande indikerar att tråden väntar på ett annat objekt, exempelvis I/O.
Övergående (Transition)
Övergående indikerar en ny tråd som väntar på nödvändiga resurser för att kunna köras.
Avslutad (Terminated)
En tråd är *avslutad* när den inte kommer at köras mer och därmed är färdig.

Den tråd som har högst prioritet bland de trådar som är *Redo* överförs till *Beredd*-läget. När den aktuella tråden har körts klart, avslutats eller på annat sätt lämnat processorn förs den tråd som ligger i *Beredd*-läget över till *Kör*-läget. Därefter kommer tråden att köras av processorn så länge som den får eller till den kommer i läget *Väntande* eller *Avslutad*. När en tråd hamnar i *Väntande*-läget kommer denna tråd att invänta de data den har efterfrågat, när de data blir tillgängliga för tråden. Därefter kommer tråden att hamna i *Redo*-läget. En tråd i läget *Avslutat* har exekverat klart och det finns ingen mer kod att köra i tråden.

När det inte finns någon tråd som är i *Redo*-läget kommer systemet att lägga in en speciell tråd kallad *Väntetråden*.

Fibrer

En fiber är en kod från användaren som schemaläggs efter en av användaren definierad algoritm. En process kan ha flera fibrer lika väl som den kan ha flera trådar, den stora skillnaden är dock att där flera trådar kan köras samtidigt kan endast en fiber köras åt gången. Detta för att underlätta portning av UNIX-applikationer som är skrivna för fiberexekveringsmodellen. Systemet skapar en fiber genom att kalla på funktionen `ConvertThredToFiber()` eller `CreateFiber()` med den stora skillnaden att `CreateFiber` inte exekverar koden innan man använder `SwitchToFiber()`-kommandot.