Systemnära programmering:

OU1: Mexec

Viktor Vikström, ens22vvm, vivi0427@student.umu.se

Programbeskrivning

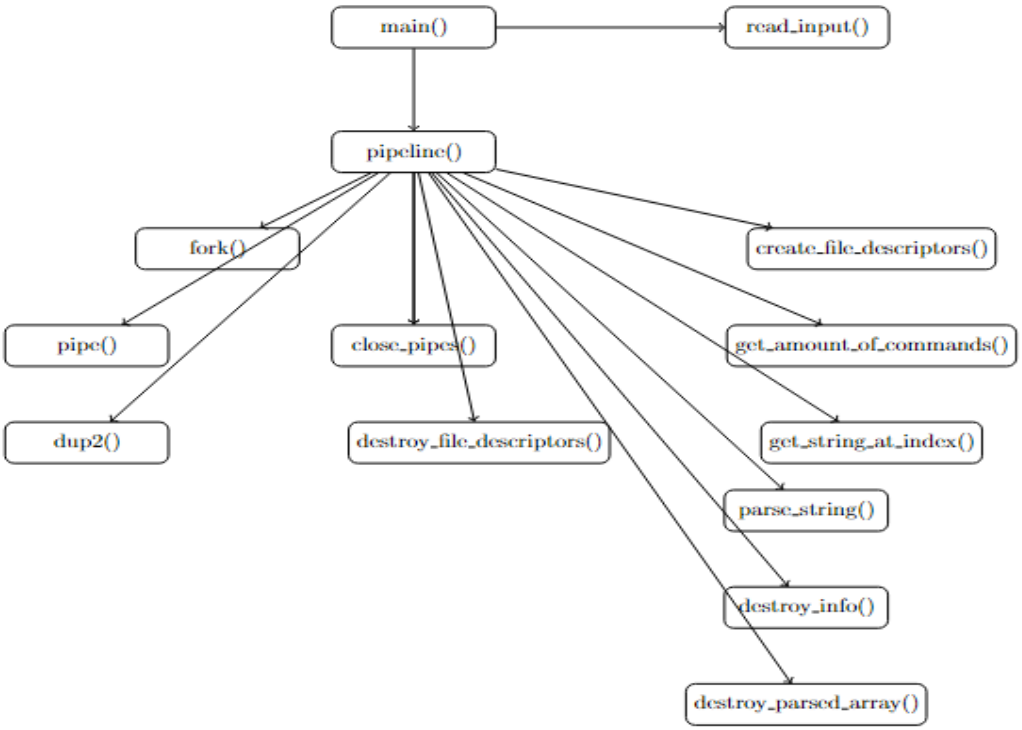
Mexec är ett program som simulerar en så kallad pipeline i skalet ”bash”. Det en pipeline gör är att via n olika processer skapar en kedja där en process utdata blir den andra processens indata. En sådan kedja skapar möjligheten att hantera data med hjälp av ett kommando för att sedan hantera dess utdata med nästa kommando.

Programmet kan hantera indata direkt ifrån terminalen där det första kommandot skrivs in först för att sedan avsluta med en ny rad tecken. Detta upprepas till alla kommandon som användaren vill använda skrivits in. För att exekvera programmet så måste tecknet EOF skrivas i terminalen.

Önskar användaren att exekvera programmet via en textfil i stället så skapas en textfil där första kommandot med dess argument placeras på första raden och det sista kommandot med dess argument placeras på den absolut sista raden i textdokumentet.

Det första som sker vid exekvering av programmet är inläsning av data som tidigare beskrivits. Detta sker via ett anrop till funktionen ”read_input”. Det ”read_input” gör först är att dynamiskt allokera en minnesplats innehållande ett heltal och ett fält av pekare till strängar med hjälp av funktionen ”create_info”.

All indata till programmet läses in rad för rad till EOF med hjälp av funktioner från biblioteket ”string.h” där varje rad innehållande ett kommando och eventuella argument lagras på en ledig plats i fältet. När all indata har blivit inläst så uppdateras heltalet till antalet kommandon som blivit inläst för att sedan returnera minnet. Det ”main” nu utför är ett anrop till funktionen ”pipeline” där en tidigare beskriven pipeline skapas. I figur 1 visas ett anropsdiagram över programmet.

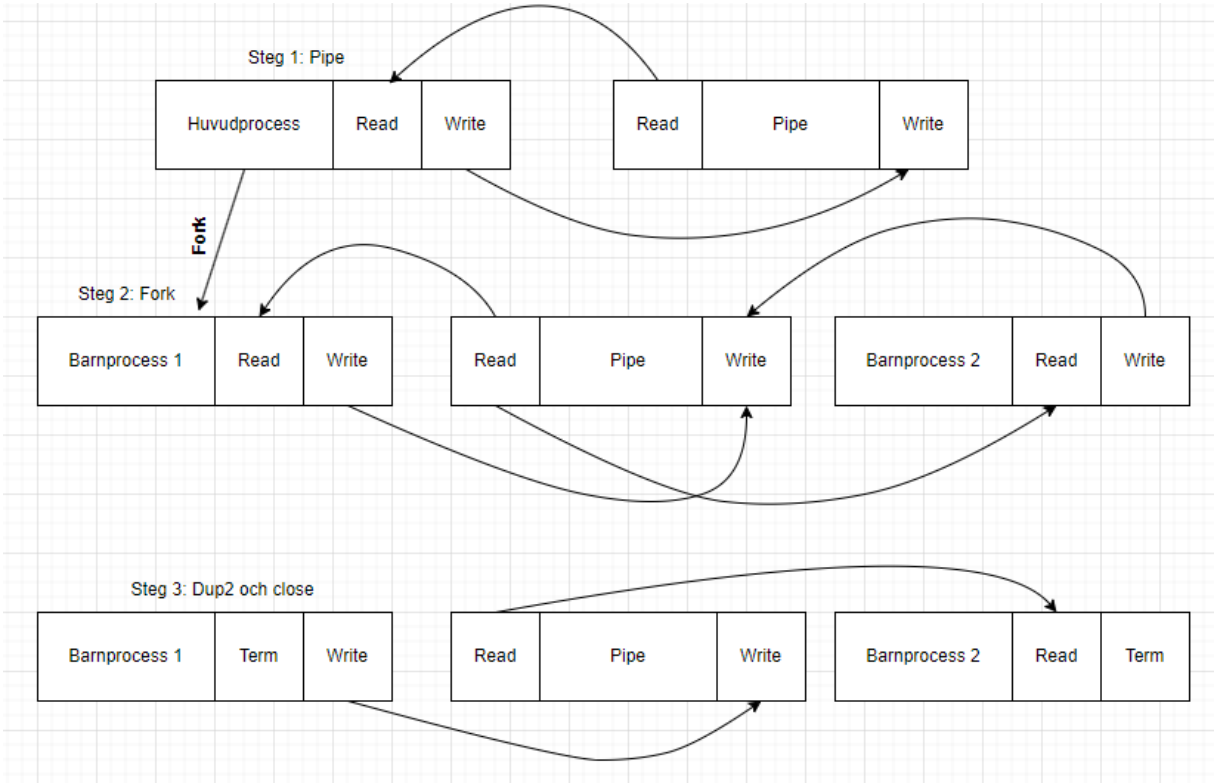


Figur 1: En visuell representation av programmets anrop.

Algoritmbeskrivning

I följande del kommer en algoritmbeskrivning till funktionen ”pipeline” som skapar processer och bygger upp pipor presenteras. Det första som funktionen ”pipeline” gör är att dynamiskt allokera minne för pipor och skapa piporna med funktionsanrop till ”pipe”. Antalet pipor som funktionen skapar är n - 1 där n är antalet kommandon.

När alla pipor har blivit skapade kommer funktionen iterera n gånger igenom följande del. En barnprocess skapas via ett funktionsanrop till ”fork”. Nu finns det två olika processer som jobbar parallellt. Barnprocessen kommer utföra två olika kontrollsatser ifall iterator-värdet är större än noll stängs skrivändan till tidigare pipa och tidigare pipa omdirigeras till barnprocessens indata. Ifall iterator-värdet är mindre än n – 1 stängs läsändan till nästa pipa och barnprocessens utdata omdirigeras till nästa pipa. Vid omdirigering så anropas funktionen ”dup2. Ett visuellt exempel på hur pipor byggs upp i ”pipeline” visas i figur 1.



Figur 1: Ett visuellt exempel på hur pipor byggs upp i funktionen ”pipeline”.

När omdirigering av in- och utdata av pipor är genomförd så stänger barnprocessen alla sina pipors läs- och skrivändor.

Barnprocessen kommer via ”execvp” anropa en specifik körbar fil. När anropet till den körbara filen är klar kommer barnprocessen att exekvera.

När ovanstående algoritm har itererat n gånger kommer huvudprocessen först stänga alla sina pipors läs- och skrivändor. För att sedan vänta in alla dem skapade barnprocesserna och kontrollera eventuella felkoder. När allt detta är gjort kommer allt dynamiskt allokerat minne städas upp och huvudprocessen exekvera.

Diskussion och reflektion

Uppgiften ”mexec” har varit ett väldigt stort hopp i svårighetsgrad inom programmering i C. Även om uppgiften har varit svår så har jag upplevt den rolig och väldigt givande. Jag har varit tvungen att nästan bara fokusera på uppgiften då den tagit väldigt mycket tid.

Det jag har upplevt svårast med uppgiften är kunskapen hur funktionen ”fork” fungerar i en loop. Detta tycker jag beror på skillnaden i svårighetsgrad på övningsuppgifterna och uppgiften ”mexec” är väldigt stor eller bara antalet övningsuppgifter.

Problem som jag har stött på under uppgiften är bland annat hanteringen av indata från användaren och öppna fildeskriptorer. Men slutsatsen med uppgiften är att den är rolig, givande och klurig men kanske lite för stor.