UMEÅ UNIVERSITET **19 september 2022**

Mexec

Systemnära programmering 5DV088 - 7,5hp

Namn och email:
Gustaf Söderlund - et14gsd@cs.umu.se

Innehåll

1 Systembeskvrining av programmet

Under laborationen har ett program skapats i programspråket C, mexec. Programmet är uppbyggt på ett sätt så att man blir ombedd att ta input antingen via standard input eller från en angiven fil. Kravet på input behöver vara ett eller flera kommandon som efter körning exekverar i en pipeline.
Varje rad representerar ett kommando. Varje ord i ett kommando representerar ett argument.
Ett exempel på hur en inputfil kan se ut åskådliggörs i figur 1.

```
cat -n mexec.c
grep -B4 -A2 return
less
```

Figur 1: inputfil

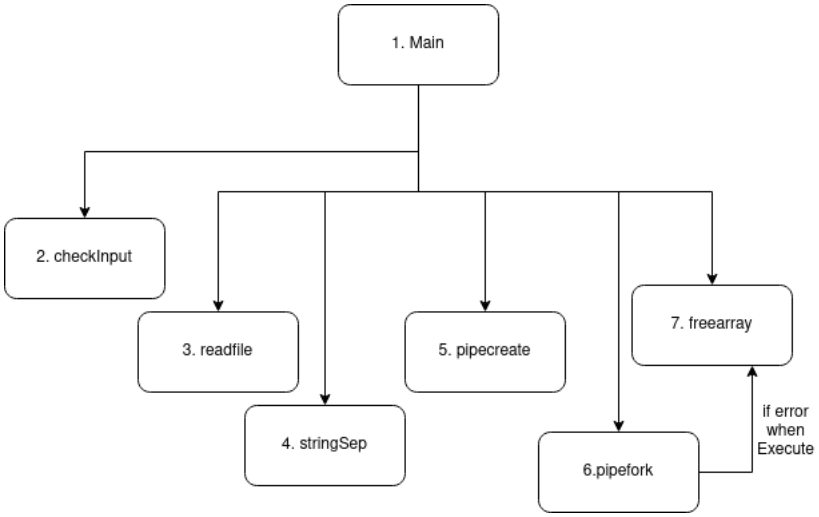
Programmet börjar med att main funktionen tar en fil som inputargument. Main kontrollerar antalet argument som ej får vara mer än två, dvs filen samt programmet, annars returneras ett felmeddelande och programmet avslutas. Main funktionen kallar på funktionen readfile som tar inputfilen och plockar ut varje rad från inputfilen och sparar det i en lista där varje element i listan är en sträng som representerar ett kommando. I readfile räknas även varje rad i den fil som angetts där antalet rader representerar antalet kommandon.

Efter att man sparat undan varje rad från filen annropas funktionen stringsep som ska splittra upp varje kommando till argument och sparar undan dem till en tvådimensionell array.

Main funktionen kallar på nästa funktion pipecreator som tar in antalet rader samt den tvådimensionella arrayen som input. I pipecreator skapas pipor som är samma antal som antalet rader i filen minus ett. Pipecreator skapar även barnprocesser med hjälp av funktionen fork(). Antalet barnprocesser ska vara samma som antalet rader i den inlästa filen. Barnprocesserna exekverar olika kommandon beroende på vilken iteration som de befinner sig i. I main väntar föräldraprocessen in barnprocesserna med hjälp av wait(), så att de hunnit exekverat. Om något fel uppstår i en barnprocess kommer ett felmeddelande skrivas ut och programmet avslutas.

Därefter kommer main funktionen kalla på sista funktionen freearray som friar det minne som tidigare allokerats.

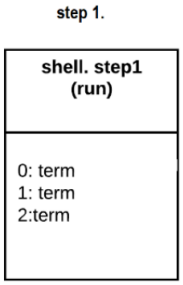
I figur 2 åskådliggörs ett annropsdiagram på hur systemet annropar de olika funktionerna som används i systemet.



Figur 2: Annropsdiagram

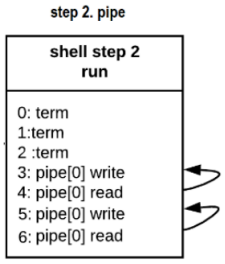
2 Algoritmbeskrivning över algoritmen för att skapa pipor

Algoritmen för den skapta funktionen pipecreator börjar med att initiera variabler och en tvådimensionell array. Om endast ett kommando anges som input ska det hanteras direkt utan att skapa några pipor eller barnprocesser. Villkor kontrolleras och kommandot exekveras genom föräldraprocessen. Misslyckas exekveringen så kommer ett felmeddelande skrivas ut och programmet avslutas. I algoritmbeskrivningen beskrivs algoritmen med tillhörande figurer. I figur 3 beskrivs start av programmet.



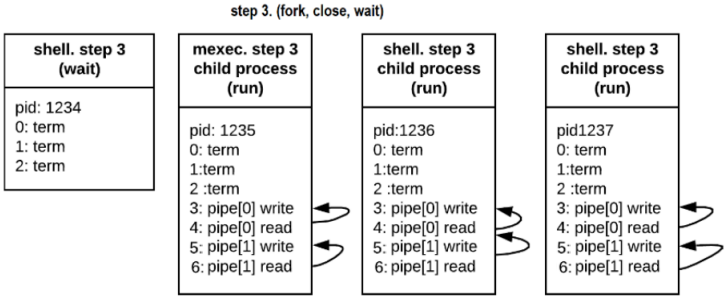
Figur 3: Start av program

En loop sätts igång som loopar lika många gånger som antalet pipor som krävs. Piporna initialiseras, skulle en pipa misslyckas att initialiseras skrivs ett felmeddelande ut och programmet avslutas vilket beskrivs i figur 4.



Figur 4: Start av program

När piporna är skapta avslutas förgående loop och en ny loop startar som loopar antalet barnprocesser som krävs. Loopen börjar med att skapa barnprocesser. Misslyckas skapandet av barnprocesser skrivs ett felmeddelande ut och programmet avslutas. I figur 5 åskådliggörs det ovannämnda.

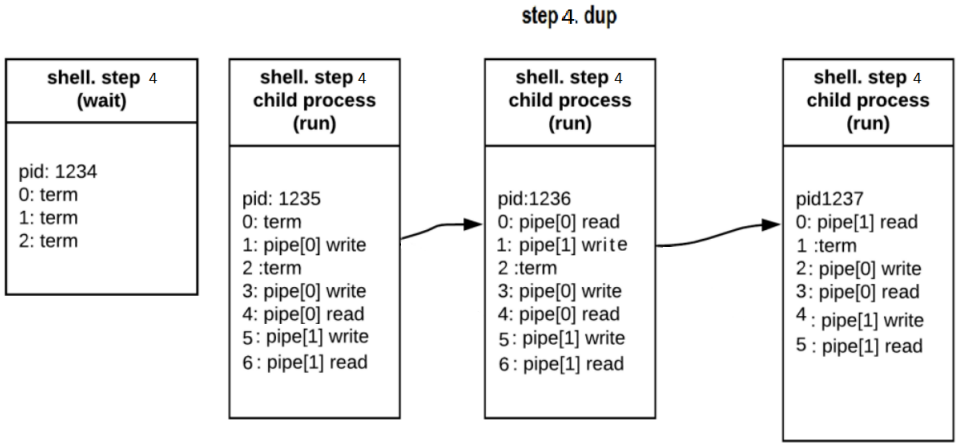


Figur 5: Start av program

Första barnprocessen som skapats kommer skriva om processens fildeskriptor så att barnprocessens output skrivs till en pipas output.

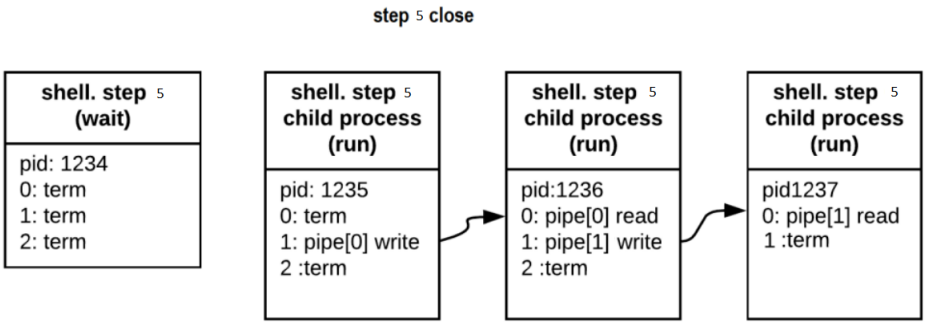
De andra barnprocesserna förutom den sista kommer att skriva om barnprocessens fildeskriptor så att input läses från den förgående pipans output och barnprocessens output skrivs till en pipas output.

Den sista barnprocessen kommer att skriva om processens fildeskripor så att input läses från den förgående pipans output, vilket åskådliggörs i figur 6.



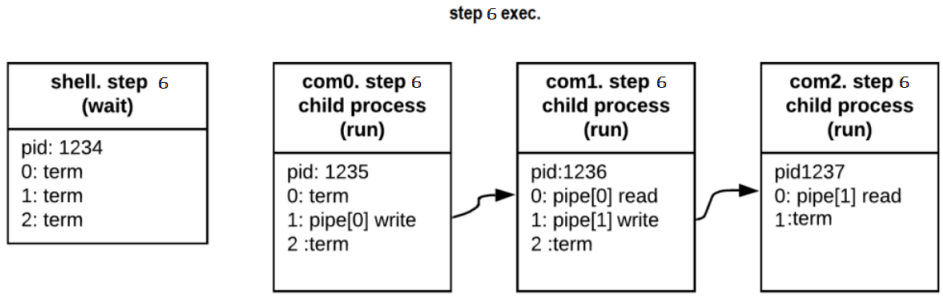
Figur 6: Start av program

Efter att processerna har skrivit om sina fildeskriptorer finns det öppna fildiskriptorer till de övriga piporna. Därför stänger man de som inte behövs, vilket beskrivs i figur 7.



Figur 7: Start av program

Det sista som sker är att exec används för att köra de program som tilldelats till respektive process, vilket åskådliggörs i figur 8.



Figur 8: Start av program

3 Diskussion och reflektion

De problem som jag stött på under laborationen var främst hur jag skulle dela upp den inlästa filen i argument och kommandon. Hade en jättebra plan på papper, men när jag skulle praktiskt implementera den så gick det inte som jag riktigt tänkt. Ett annat problem jag stötte på under laborationen som jag försökt lösa under sommarstödet var att mina utskrifter i terminalen som försvann när man skulle ange input själv och inte via en fil. Visade sig att felet var att jag lyckades få programmet att crasha när man matade in input via stdin.

Jag tycker laborationsuppgiften är väldigt bra då man får lära sig massor av nya verktyg samt att jag har blivit bättre på att söka efter dokumentation om olika funktioner i C på nätet.

4 Referenser

Figurerna till algoritmbeskrivningen är inspirerad från en gammal tenta i kursen. Figurerna är dock handgjorda efter mitt program i paint och diagrams.net.