IT University of Copenhagen

Obligatorisk Opgave 1

Operativsystemer og C

Omar Khan (omsh@itu.dk)
Mads Ljungberg (malj@itu.dk)
02/10/2015

Formål

Formålet med denne opgave er at lære og bruge koncepterne omkring processer i operativsystemer, redirection af input og output, og pipe i et shell program skrevet i C. Forud for opgaven er vi blevet tildelt oo1.zip, der indeholder kildekoden som vi skal arbejde ud fra.

Baggrund

Ifølge opgavebeskrivelsen skal programmet kunne opfylde følgende specifikationer:

- Uafhængighed: Programmet skal kunne virke uden brug af andre shells f.eks. ved at benytte et systemkald `system()` til at starte en bash.
- Når programmet kører skal det vise navnet på den host der er logget ind.
- Udskrive en "Command not found" meddelelse når man skriver en kommando der ikke findes i systemet.
- Kommandoer skal kunne køres i baggrunden, ved brug af `'&'`. Dvs. at man skal kunne fortsætte med at benytte `bosh`.
- Man skal kunne lave redirection af `stdin` og `stdout` ved at benytte, `'<'` og `'>'`.
- Det skal være muligt at anvende pipes. Dvs. at det skal være muligt at tage flere kommandoer med `'|'` som separator. Den venstre specificeret kommando skal benyttes som input til den højre specificeret kommando.
- Man skal kunne lukke shellen ved at bruge kommandoen `'exit'`.
- Ctrl+C skal ikke lukke shellen, men afslutte det kørende program i `bosh`.

De udleverede filer indeholder tre C-filer:

- bosh.c
- parser.c
- print.c

For at løse opgaverne specificeret er der ikke behov for at ændre i parser.c og print.c.
I filen parser.h er der specificeret de to `struct` som bliver benyttet i programmet, `Cmd` og `Shellcmd`.

`Cmd` har en hægtet liste datastruktur, der består af en kommando streng og peger på den næste kommando i rækken. Når der er flere kommandoer skal der anvendes pipe.

`Shellcmd` indeholder felter relateret til de andre specificeret opgaver, som hvis brugeren har specificeret at programmet skal køres i baggrunden, eller specificeret en form af redirect.

Implementation

Hostname

I bosh.c er der en metode signatur, `gethostname()`, til at hente brugernavnet. For at finde lokationen af hostname, benyttes det virtuelle filsystem `/proc`.
Den fulde sti til hostname filen er `proc/sys/kernel/hostname`.
Vi har lavet den antagelse, at vi kun skal hente den første linje i filen for at få hostname.
Dette har vi gjort ved at benytte en `FILE` og metoden `fopen()`.

```
FILE *fp;
  char *line = NULL;
  size_t len = 0;
  ssize_t read;

  fp = fopen(hostfile,"r");
  if((read = getline(&line, &len, fp)) != -1) /* get the
hostname from line 1 */
  {
    strtok(line,"\n"); /* remove newline token */
    *hostname = line;
  }
```

Baggrundsprocesser & Redirection

Implementering af baggrundsprocesser og redirection er gjort i metoden `executecmds()` i filen execmds.c.
Metoden tager imod en kommando, `Cmd`, et filnavn til `stdin`, et filnavn til `stdout`, og en binær boolean for at vide om programmet skal køre i baggrunden.
Den observante læser har bemærket, at metodens parametre er svarende til `Shellcmd`'s felter.
Metoden bliver kaldt fra bosh.c's `executeshellcmd()`,

```
executecmds(cmds, shellcmd->rd_stdin, shellcmd->rd_stdout,
shellcmd
```

Metoden skaber en ny process hvori den tjekker om parametrene er instantieret.

```
...
if(pid == 0)
  {
    if(infilename != NULL) { ... }

    if(outfilename != NULL) { ... }
    ...
  }
  else if(bg != 1)
  {
    waitpid(pid, &status, 0);
  }
...
```

Til redirection er der gjort brug af metoderne `open()`, `close()` og `dup()`.
Hvis programmet skal køres i baggrunden ventes der ikke på processen.

Pipe

I metoden `executecmds()` tjekkes `Cmd` struct'en om der er flere kommandoer og hvis dette er tilfældet kaldes metoden `pipecmd()` i filen pipe.c. Metoden tager en kommando som argument. Metoden er rekursiv og derfor startes der med at tjekke om kommandoen er `NULL`. Hvis det ikke er tilfældet oprettes en `pipe()` og der startes en ny proces.

```
  ...
  if(cmds != NULL)
  {
    pipe(pfd); /* Create the pipe */

    if((pid = fork()) == 0) /* Child */
    {
        ...
       execvp(*cmd, cmd);
     }
    else /* Parent */
    {
      ...
      pipecmd(nextcmds);
    }
  }
  ...
```

I den nye proces bliver pipe's `stdin` udskiftet med processens `stdin` ved brug af `dup2()`. Tilbage i den gamle proces udskiftes `stdout` på samme vis.

Ctrl+C

Når en bruger trykker Ctrl+C sendes der et signal som kan fanges af det program der bliver kørt.
Helt specifikt er signalet for Ctrl+C en `SIGINT`. Når et signal fanges med metoden `signal()`, skal man i dens andet argument angive en metode der skal kaldes og i dette tilfælde er det sat til en tom metode, `sig_handler()`. Dette giver den ønskede funktionalitet.

```
#include <signal.h>
...
void sig_handler(int signo) { }

int main(int argc, char *argv[])
{
...
  signal(SIGINT, sig_handler);
...
```

Funktioner som "exit" kommandoen og "Command not found" beskeden er også implementeret i bosc.c.

Test

Vi har manuelt testet programmet, hvilket betyder, at der er stor sandsynlighed for at alle test cases ikke er blevet dækket.
Eksemplerne fra opgavebeskrivelsen er blevet afprøvet og fungerer efter hesigten. Under afprøvelser af diverse programmer fandt vi problemer med hensyn til indtastning af en forkert kommando. Programmet printer efter intentionen "Command not found", men af en årsag vi ikke er kommet frem til kan man ikke bare lukke bosh ved at bruge "exit". Man skal nemlig udføre "exit" for antallet af gange man har skrevet en forkert kommando. Antagelsen er, at der startes en process der ikke bliver stoppet korrekt.

Konklusion

Ud fra specifikationerne til opgaven kan der konkluderes, at `bosh` kan køre programmer i baggrunden, redirect til filer, kører flere programmer ved brug af pipe og er uafhængig.