Deluppgift 14 Systemanrop för tangentbord och skärm: read och write

Beskrivning av uppgiftens systemanrop

Systemanropen read och write är deklarerade i `lib/user/syscall.h` och används av användarprogrammen därefter:

`int read (int fd, void *buffer, unsigned length);`
Läser `length` tecken från resursen `fd` och lagrar i `buffer`. Returnerar antal lästa tecken. *I denna deluppgift hanteras endast* `fd` == `STDIN_FILENO` *(tangentbord).* Alla andra `fd` betraktas som fel och ger returvärde -1.

`int write (int fd, const void *buffer, unsigned length);`
Skriver `length` tecken lästa från `buffer` till resursen `fd`. Returnerar antal skrivna tecken. *I denna deluppgift hanteras endast* `fd` == `STDOUT_FILENO` *(skärm).* Alla andra `fd` betraktas som fel och ger returvärde -1.

Saker som kan vara bra att känna till

Systemanropet `read` används för att läsa från både tangentbortet och från en öppen fil. Systemanropet `write` används både för att skriva till skärmen och till en öppen fil. För att avgöra var läsning/skrivning skall ske används en fildeskriptor, `fd` (första parametern). Det är ett heltal som representerar en öppen fil i kernel. För skärm och tangentbord reserveras två speciella `fd`. De är definerade enligt föjlande i filen `lib/stdio.h`

```
#define STDIN_FILENO 0
#define STDOUT_FILENO 1
```

Om `fd` är lika med `STDIN_FILENO` skall anrop av `read` alltså läsa från tangentbordet. Till detta kan du använda funktionen `input_getc` som läser ett tecken från tangentbordet. Den finns deklarerad i `devices/input.h` och är implementerad i motsvarande c-fil. Tänk på att användaren normalt brukar vilja se varje inmatat tecken på skärmen medan denne skriver. Om `STDIN_FILENO` anges vid anrop av `write` hanteras det som ett fel genom att returnera felkoden minus ett (-1).[1]

Om `fd` är lika med `STDOUT_FILENO` skall anrop av `write` skriva till skärmen. Till detta kan du använda funktionen `putbuf` som finns deklarerad i `lib/kernel/stdio.h` och implementerad i `lib/kernel/console.c`. Om `STDOUT_FILENO` anges vid anrop av `read` hanteras det som ett fel genom att returnera felkoden minus ett (-1).[2]

Observera att det är `putbuf` som skriver till användarens skärm (det svarta fönstret som kommer upp om du startar PintOS utan flaggan -v), inte `printf`. Funktionen `printf` är endast menad att skriva information till dig som programmerare, som visas i den terminal som via serieporten är ansluten till PintOS-datorn under utvecklingen (terminalen du startar PintOS i).

---

1. Jag känner inte till några tangentbord man kan skriva till, men den uppfinningsrike kan ju bygga ett tangentbord där varje tangent har en liten motor som gör att tangenten åker ned när datorn skriver dess tecken till tangentbordet. Fast det är ju inte helt otroligt att det redan är gjort för någon filminspelning där något osynligt fenomen skall använda ett tangentbord.
2. Det finns visserligen pekskärmar som ju bevisligen tar emot indata från användaren, men då kan operativsystemet internt hantera det som två enheter, en skärm (write only) och en inmatningsenhet (read only).

Både `read` och `write` tar emot en minnesbuffer[1] (andra parametern) med specificerad storlek (tredje parametern). Systemanropet read skall placera de lästa tecknen i buffern tills den är full. Det är viktigt att aldrig skriva fler tecken till buffern än storleken anger (så klart). Systemanropet write fungerar tvärtom, läser tecken att skriva ut från angiven buffer. Det är viktigt att skriva ut exakt alla tecken i buffern (så klart). Både read och write returnerar antalet behandlade tecken. Ofta är returvärdet för read och write identiskt med tredje inparametern, om inte något fel uppstod.

Att läsa inmatning från användaren brukar vara lite speciellt. Normalt kan användaren skriva indata till programmmet, och ångra sig och ändra så länge inmatningen inte bekräftats med “Return” eller “Enter”. Först när bekräftelsen kommer får programmet tillgång till den rad som matats in. Dessutom brukar programmet då inte vänta på fler tecken (för att fylla hela buffern) utan nöja sig med det som matats in. Slutligen krävs det i C att text-stängar avslutas med ett noll-tecken. Tillsammans skulle detta innebära att tredje parametern anger hur många tecken i buffern som *maximalt* får användas, inklusive avslutande noll-tecken (`'\0'`). I PintOS antar vi att dessa speciella detaljer tas om hand av en getline-funktion i ett programbibliotek, och inte av operativsystemets systemanrop.
Systemanropet skall alltså läsa exakt ***så många tecken som anges av andra parametern. Avslutande noll-tecken är inte heller systemanropets bekymmer i vårt fall.***

I PintoOS finns ytterligare en speciell sak. Denna skall systemanropet lösa. Om användaren matar in ett carriage-return tecken (`'\r'`) skall detta betraktas som, och ersättas med, ett nyradstecken (`'\n'`). Detta för att “Enter” eller “Return” i PintOS endast ger carriage-return tecken, medan vi när vi programmerar i UNIX normalt förväntar oss ett nyradstecken i detta läge.

Uppgift

Implementera systemanropen `read` och `write` för läsning från tangentbordet respektive skrivning till skärm. *Filer hanteras senare.* Observera att tangentbordet och skärmen är resurser som delas av alla trådar och processer. Fundera på hur det fungerar om två olika processer vill använda tangentbordet samtidigt.

Testa dina systemanrop med ett enkelt testprogram som läser rader från tangentbordet och skriver ut dem igen. *Var noga med att skriva indata i terminalen,* inte i det emulerade fönstret (det emulerade fönstret har inte rätt tangentbordslayout). Utdata skall visas i ***bägge*** fönstren. Naturligtvis vill användaren kunna se de tecken han/hon matar in på tangentbordet direkt på skärmen.

Filen `examples/line_echo.c` innehåller ett lämpligt testprogram. Filen `examples/file_syscall_tests.c` innehåller fler tester, de första för denna uppgift, och de senare för kommande uppgifter. Tänk på att `file_syscall_tests` är ett för långt filnamn för PintOS, se kommentar i filen för hur du kan starta användarprogrammet.

1. En “buffer” är en array med en viss storlek att lagra eller hämta data ur.