Lösning dugga 2022-06-21

Totalt 0.4 p

Skriv/klistra in svar direkt i detta dokument!

Tillåtna redskap:

Valfri Linuxterminal, Nano, Visual Studio Code, Notepad samt Notepad++.

1. Nedanstående program används för att generera en valbar fördröjning mätt i millisekunder via en körbar fil döpt *main*. Programmet skall vid körning kunna medföra en fördröjning på 1 sekund via följande kommando:

$ delay 1000

Filen *main.c:*

```c
/* Inkluderingsdirektiv: */
#include <unistd.h>
#include <stdlib.h>

/*****************************************************************************
* main: Genererar fördröjning via funktionen usleep från biblioteket unistd.h.
*****************************************************************************/
int main(const int argc, const char** argv)
{
   usleep(atoi(argv[0]) * 1000);
   return 0;
}
```

a) Vid körning genereras ingen fördröjning. Ett felmeddelande skrivs inte heller ut. Förklara varför ingen fördröjning sker samt korrigera så att programmet fungerar som det ska och ett felmeddelande skrivs ut ifall ingen fördröjningstid är angiven. Klistra in korrigerad kod.

(0.05 p)

Kommentar:

Det primära problemet är att första ingående argument argv[0] består av aktuellt kommando för att köra programmet. Eftersom returvärdet från funktionen atoi är 0 så innebär detta att kommandot består enbart eller delvis av bokstäver (atoi kan typomvandla en sträng som börjar på ett tal till motsvarande heltal, exempelvis typomvandlas 21Ela till 21). I övriga fall blir returvärdet 0, vilket innebär att fördröjningen i detta fall blir 0.

Fördröjningstiden skall passeras som andra ingående argument, alltså argv[1]. Dessutom bör kontroll göras att minst två ingående argument har matats in, annars bör felkod 1 returneras och programmet avslutas direkt.

Korrigerad kod:

```c
/* Inkluderingsdirektiv: */
#include <unistd.h>
#include <stdlib.h>

/*****************************************************************************
* main: Genererar fördröjning via funktionen usleep från biblioteket unistd.h.
*****************************************************************************/
int main(const int argc, const char** argv)
{
   if (argc != 2) return 1;
   const size_t delay_time = (size_t)atoi(argv[1]);
   usleep(delay_time * 1000);
   return 0;
}
```

b) Antag att ovanstående program är sparat i en fil döpt *main.c.* Skapa en makefil som möjliggör enskild kompilering och exekvering av programmet via nedanstående kommandon (default-target skall läggas till i uppgift 2, så tänk inte på det här):

$ make build *# Kompilerar filen main.c.*
$ make delay time=500 *# Kör programmet med en fördröjningstid på 500 ms.*

(0.05 p)

Makefilens innehåll:

```
#########################################################################
# build: Kompilerar filen main.c och skapar en körbar fil döpt main.
#########################################################################
build:
        @gcc main.c -o main -Wall

#########################################################################
# delay: Kör ett fördröjningsprogram via den körbara filen main,
#        där ingående argument time utgör av fördröjningstiden
#        mätt i millisekunder.
#########################################################################
delay:
        @./main $(time)
```

2. Skapa ett shell-skript döpt *blink.sh*, där användaren vid körning av skriptet skall ange PIN-nummer för två lysdioder, blinkhastighet i millisekunder samt antalet blinkningar som ingående argument till shell-skriptet för GPIO-implementering med *libgpiod.* Blinkningarna skall genomföras enligt följande:

LED1 samt LED2 tänds.
Fördröjning.
LED1 släcks.
Fördröjning.
LED2 släcks.
Fördröjning.

a) Skriv shell-scriptet *blink.sh* nedan. Använd den körbara filen från uppgift 1 för att generera specificerad fördröjning via make. Ni skall alltså INTE använda flaggan *–mode=time* för att implementera fördröjning!

(0.1 p)

Shell-scriptet *blink.sh:*

```
#!/bin/bash

#########################################################################
# blink: Blinkar två lysdioder anslutna till PIN $1 och $2 med
#        en fördröjningstid på $3 millisekunder. Lysdioderna går
#        från att vara tända till att släckas en efter en.
#########################################################################
blink()
{
   gpioset 0 $1=1 $2=1
   make delay time=$3
   gpioset 0 $1=0 $2=1
   make delay time=$3
   gpioset 0 $1=0 $2=0
   make delay time=$3
   return 0
}
```

```
#########################################################################
# main: Genomför blinkning av två lysdioder anslutna till PINs
# $1-$2 med en blinkhastighet på $3 ms, vilket genomförs $4 gånger
# via anrop av funktionen blink.
#########################################################################
main()
{
   for ((i = 0; i < $4; ++i))
   do
      blink $1 $2 $3
   done
}

# Anropar funktionen main vid start, passerar fyra parametrar från terminalen:
main $1 $2 $3 $4
```

b) Uppdatera makefilen från uppgift 1 så att shell-skriptet *blink.sh* kan köras ett target döpt *blink*, där användaren skall kunna skriva följande för att blinka två lysdioder anslutna till PIN 17 och 22 med en blinkhastighet på 100 ms 20 gånger:

$ make blink led1=17 led2=22 delay_time=100 blinks=20 *# Kör shell-scriptet blink.sh.*

Skapa också ett default-target i makefilen, som medför att C-koden från uppgift 1 kompileras följt av att shell-scriptet *blink.sh* körs ifall inget target anges.

(0.05 p)

Makefilens innehåll (med targets *blink* samt *all):*

```
#########################################################################
# all: Kompilerar C-kod till ett fördröjningsprogram och kör sedan
#      ett script för blinkning av två lysdioder, där detta
#      fördröjningsprogram används för att generera blinkningarna.
#########################################################################
all: build blink

#########################################################################
# build: Kompilerar filen main.c och skapar en körbar fil döpt main.
#########################################################################
build:
        @gcc main.c -o main -Wall

#########################################################################
# delay: Kör ett fördröjningsprogram via den körbara filen main,
#        där ingående argument time utgör av fördröjningstiden
#        mätt i millisekunder.
#########################################################################
delay:
        @./main $(time)

#########################################################################
# blink: Kör script för blinkning av två lysdioder valbart antal
#        gånger med valbar fördröjningstid mätt i millisekunder.
#########################################################################
blink:
        @sudo bash blink.sh $(led1) $(led2) $(delay_time) $(blinks)
```

3. I C-programmet på nästa sida används *libgpiod* för att blinka tre lysdioder lagrade i en statisk array, där blinkhastigheten samt antal blinkningar anges från terminalen.

a) Programmet kompilerar, men när programmet körs blinkar inte lysdioderna. Kontrollera programmet och korrigera eventuella fel direkt i koden. Implementera inline-kommentarer som förklarar vad som var fel, alternativt placera dessa kommentarer i befintliga kommentarsblock. Klistra in korrigerad kod längst bak i detta dokument!

Redogör kortfattat varför programmet inte fungerar och ange hur koden kan göras tydligare för att minska risken för framtida fel samt göra koden mer lättförståelig i allmänhet.

(0.1 p)

Kommentarer, motiveringar och synpunkter gällande korrigeringar av C-koden:

Filsökvägen till gpiochip0 är fel, det stod gpiochip0, men korrekt är /dev/gpiochip0. Dessutom stängs gpio-chippet efter att en ny GPIO-linje har initierats. Ingående argument som skall läsas av är arvg[1] till argv[5] och sex argument skall därmed läsas in (argv[0] är enbart kommandot för att köra programmet och ignoreras). I de nästlade for-satserna räknades iterator i upp i stället för j i den inre for-satsen.

Koden kan göras tydligare genom att använda fler objekt i stället för att hålla koden så komprimerad som det är i detta fall, vilket gör det svårt att följa. Exempelvis kan konstanter skapas för samtliga ingående argument, såsom pin1, pin2, pin3, delay_time samt blinks, för att göra koden mer lättförståelig.

b) Skapa en makefil för projektet, som möjliggör enskild samt kombinerad kompilering och körning av programmet, där PIN-numren för tre lysdioder, fördröjningstiden samt antalet blinkningar kan specificeras via ingående argument enligt nedan:

$ make led1=17 led2=22 led3=23 delay_time=100 blinks=200

Utskriften i terminalen skall då se ut enligt nedan:

led1 at PIN 17 configured as output!
led2 at PIN 22 configured as output!
led3 at PIN 23 configured as output!

Se till att kompilera med *libgpiod.* Kommandon som utförs när make används skall ej skrivas ut i terminalen. Kompilera enbart använda C-filer. Det skall alltså vara möjligt att aktuell katalog innehåller C-filer som inte ingår i projekt (även om detta är dålig kutym).

(0.05 p)

Makefilens innehåll:

```
#####################################################################
# all: Kompilerar C-kod och kör C-program för GPIO-implementering.
#####################################################################
all: build run

#####################################################################
# build: Kompilerar filen main.c och skapar en körbar fil döpt main.
#####################################################################
build:
	@gcc main.c -o main -Wall -l gpiod

#####################################################################
# run: Kör script för blinkning av två lysdioder valbart antal
#      gånger med valbar fördröjningstid mätt i millisekunder.
#####################################################################
run:
	@./main $(led1) $(led2) $(led3) $(delay_time) $(blinks)
```

Filen *main.c:*

```c
/* Inkluderingsdirektiv: */
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <gpiod.h>
#include <unistd.h>

/* Enumerationer: */
enum gpio_direction { GPIO_DIRECTION_IN, GPIO_DIRECTION_OUT };

/*****************************************************************************
* gpiod_line_new: Returnerar ny GPIO-linje, som initieras till in- eller utport.
*****************************************************************************/
static struct gpiod_line* gpiod_line_new(const uint8_t pin, const enum gpio_direction direction,
                                         const char* alias)
{
   struct gpiod_chip* chip0 = gpiod_chip_open("gpiochip0");
   struct gpiod_line* self = gpiod_chip_get_line(chip0, pin);

   if (direction == GPIO_DIRECTION_OUT)
   {
      gpiod_line_request_output(self, alias, 0);
      printf("%s at PIN %u configured as output!\n", gpiod_line_consumer(self),
             gpiod_line_offset(self));
   }
   else
   {
      gpiod_line_request_input(self, alias);
      printf("%s at PIN %u configured as input!\n", gpiod_line_consumer(self),
             gpiod_line_offset(self));
   }

   gpiod_chip_close(chip0);
   return self;
}

/*****************************************************************************
* main: Ansluter tre lysdioder, vars PIN-nummer matas in från terminalen.
*       För respektive lysdiod initieras en GPIO-linjepekare, som lagras i en array.
*       Blinkhastigheten samt antalet blinkningar matas in från terminalen.
*       Lysdioderna blinkar sedan specificerat antal gånger.
*****************************************************************************/
int main(const int argc, const char** argv)
{
   if (argc != 6) return 1;

   struct gpiod_line* leds[] =
   {
      gpiod_line_new((uint8_t)atoi(argv[0]), GPIO_DIRECTION_OUT, "led1"),
      gpiod_line_new((uint8_t)atoi(argv[1]), GPIO_DIRECTION_OUT, "led2"),
      gpiod_line_new((uint8_t)atoi(argv[2]), GPIO_DIRECTION_OUT, "led3")
   };

   for (size_t i = 0; i < (size_t)atoi(argv[4]); ++i)
   {
      for (struct gpiod_line** j = leds; j < leds + sizeof(leds) / sizeof(struct gpiod_line*); ++i)
      {
         gpiod_line_set_value(*j, 1);
         usleep((size_t)atoi(argv[3]) * 1000);
         gpiod_line_set_value(*j, 0);
      }
   }

   return 0;
}
```

Korrigerad kod:

```c
/* Inkluderingsdirektiv: */
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <gpiod.h>
#include <unistd.h>

/* Enumerationer: */
enum gpio_direction { GPIO_DIRECTION_IN, GPIO_DIRECTION_OUT };

/*****************************************************************************
* gpiod_line_new: Returnerar ny GPIO-linje, som initieras till in- eller utport.
*****************************************************************************/
static struct gpiod_line* gpiod_line_new(const uint8_t pin, const enum gpio_direction direction,
   const char* alias)
{
   struct gpiod_chip* chip0 = gpiod_chip_open("/dev/gpiochip0"); // Korrigerade filsökvägen.
   struct gpiod_line* self = gpiod_chip_get_line(chip0, pin);

   if (direction == GPIO_DIRECTION_OUT)
   {
      gpiod_line_request_output(self, alias, 0);
      printf("%s at PIN %u configured as output!\n", gpiod_line_consumer(self),
         gpiod_line_offset(self));
   }
   else
   {
      gpiod_line_request_input(self, alias);
      printf("%s at PIN %u configured as input!\n", gpiod_line_consumer(self),
         gpiod_line_offset(self));
   }

   // Tog bort stängning av gpiochip0.
   return self;
}
```

```c
/*****************************************************************************
* main: Ansluter tre lysdioder, vars PIN-nummer matas in från terminalen.
*       För respektive lysdiod initieras en GPIO-linjepekare, som lagras i en array.
*       Blinkhastigheten samt antalet blinkningar matas in från terminalen.
*       Lysdioderna blinkar sedan specificerat antal gånger.
*****************************************************************************/
int main(const int argc, const char** argv)
{
   if (argc < 6) return 1; // Bytte från 5, returnerar felkod 1 om för få argument har matats in.

   // Tydliggör koden via konstanter:
   const uint8_t pin1 = (uint8_t)atoi(argv[1]);
   const uint8_t pin2 = (uint8_t)atoi(argv[2]);
   const uint8_t pin3 = (uint8_t)atoi(argv[3]);
   const size_t delay_time = (size_t)atoi(argv[4]);
   const size_t blinks = (size_t)atoi(argv[5]);

   struct gpiod_line* leds[] =
   {
      gpiod_line_new(pin1, GPIO_DIRECTION_OUT, "led1"),
      gpiod_line_new(pin2, GPIO_DIRECTION_OUT, "led2"),
      gpiod_line_new(pin3, GPIO_DIRECTION_OUT, "led3")
   };

   const size_t num_leds = sizeof(leds) / sizeof(struct gpiod_line*);

   for (size_t i = 0; i < blinks; ++i)
   {
      for (struct gpiod_line** j = leds; j < leds + num_leds; ++j) // Bytte från ++i.
      {
         gpiod_line_set_value(*j, 1);
         usleep(delay_time * 1000);
         gpiod_line_set_value(*j, 0);
      }
   }

   return 0;
}
```