

Lösning repetitionsdugga 2022-06-02

Namn: _____

Totalt 0.2 poäng, varav

- ✓ **0.1 p för korrekt svar på majoriteten av frågorna**
- ✓ **0.2 p för korrekt svar på samtliga frågor**

Bifoga svar direkt i detta dokument!

1. Du har en lysdiod ansluten till PIN 17 samt en tryckknapp ansluten till PIN 27, som skall implementeras i ett C-program med det virtuella filsystemet sysfs. För att kontrollera att kopplingarna är korrekta så skall du testa att tända lysdioden samt läsa av tryckknappens insignal, både vid nedtryckning samt i normalläge. Ange Linuxkommandon för att genomföra följande:

- a) Reservera PIN 17 samt 27 för aktuell process. Sätt PIN 17 till utport samt PIN 27 till inport.

```
$ echo 17 > /sys/class/gpio/export
$ echo 27 > /sys/class/gpio/export
$ echo out > /sys/class/gpio/gpio17/direction
$ echo in > /sys/class/gpio/gpio27/direction
```

- b) Tänd sedan lysdioden.

```
$ echo 1 > /sys/class/gpio/gpio17/value
```

- c) Läs därefter av tryckknappens insignal. Vad bör du kunna avläsa vid nedtryckt respektive icke nedtryckt tryckknapp?

```
$ cat /sys/class/gpio/gpio27/value
```

Vid nedtryckt tryckknapp bör signalen vara hög (1), annars låg (0).

- d) Släck lysdioden.

```
$ echo 0 > /sys/class/gpio/gpio17/value
```

- e) Ta bort reservation av PIN 17 samt 27.

```
$ echo 17 > /sys/class/gpio/unexport
$ echo 27 > /sys/class/gpio/unexport
```

OBS! Vänd blad!

Hårdvarunära programmering

2. I följande program skall en dynamisk vektor fyllas med 15 osignerade tal 100 – 114.

Dessa tal skall sorteras i fallande ordning, alltså från 114 ned till 100, och sedan skrivs till en fil döpt *vector.txt*. Innehållet från denna fil skall sedan läsas in rad för rad och skrivs ut i konsolen. Programmet är dock inte slutfört.

- a) Slutför samtliga vektorfunktioner i filen *vector.c* samt funktionen *file_read* i filen *main.c*.

Skriv kompletterande programkod direkt i funktionerna på efterföljande sidor!

- b) Efter att ha slutfört programmet skall detta testas i Ubuntu. Ange Linuxkommandon för att genomföra följande:

- Skapa en ny katalog döpt *vector_example* och dirigera till denna.

```
$ mkdir vector_example
$ cd vector_example
```

- Öppna filerna *header.h*, *main.c* samt *vector.c* i katalogen *vector_example* med radnummer i nano-editorn.

```
$ sudo nano header.h -l
$ sudo nano main.c -l
$ sudo nano vector.c -l
```

- Kompilera dessa filer med GCC-kompilatorn och skapa en körbar fil döpt *main*.

```
$ gcc *c -o main -Wall
```

- Köra programmet från terminalen.

```
./main
```

Filen *header.h*:

```
#ifndef HEADER_H_
#define HEADER_H_

/* Inkluderingsdirektiv: */
#include <stdio.h>
#include <stdlib.h>

/* Enumerationer: */
enum sort_direction { SORT_DIRECTION_ASCENDING, SORT_DIRECTION_DESCENDING };

/*****
 * vector: Dynamisk vektor för lagring av osignerade heltal.
 *****/
struct vector
{
    size_t* data; /* Pekare till dynamiskt fält innehållande osignerade heltal. */
    size_t size; /* Antalet element i fältet. */
};

/* Funktionsdeklarationer: */
void vector_new(struct vector* self, const size_t size);
void vector_assign(struct vector* self, const size_t start_val);
void vector_sort(struct vector* self, const enum sort_direction sort_direction);
void vector_print(const struct vector* self, FILE* stream);

#endif /* HEADER_H_ */
```

Filen *main.c*:

```
/* Inkluderingsdirektiv: */
#include "header.h"

/*****
 * file_read: Läser varje rad från en fil en efter en och skriver ut i konsolen.
 *****/
static void file_read(const char* filepath)
{
    FILE* fstream = fopen(filepath, "r");
    char s[100];

    if (!fstream)
    {
        fprintf(stderr, "Could not open file at path %s!\n\n", filepath);
    }
    else
    {
        s[0] = '\0';
        while (fgets(s, sizeof(s), fstream))
            fprintf(stdout, "%s", s);
        fclose(fstream);
    }

    return;
}

/*****
 * main: Fyller en dynamisk vektor / array med 15 osignerade heltal 100 - 114.
 *       Innehållet sorteras i fallande ordning och skrivs till filen vector.txt.
 *       Innehållet från filen läses sedan in rad för rad och skrivs ut i konsolen.
 *****/
int main(void)
{
    struct vector v;
    vector_new(&v, 15);
    vector_assign(&v, 100);
    vector_sort(&v, SORT_DIRECTION_DESCENDING);

    FILE* fstream = fopen("vector.txt", "w");
    vector_print(&v, fstream);
    fclose(fstream);

    file_read("vector.txt");
    return 0;
}
```

Filen *vector.c*:

```

/* Inkluderingsdirektiv: */
#include "header.h"

/*****
 * vector_new: Initieringsrutin för objekt av strukten vector.
 *****/
void vector_new(struct vector* self, const size_t size)
{
    self->data = (size_t*)malloc(sizeof(size_t) * size);
    if (!self->data) self->size = 0;
    else self->size = size;
    return;
}

/*****
 * vector_assign: Fyller dynamisk vektor med osignerade heltal i stigande ordning.
 *****/
void vector_assign(struct vector* self, const size_t start_val)
{
    size_t num = start_val;
    for (register size_t* i = self->data; i < self->data + self->size; ++i)
        *i = num++;
    return;
}

/*****
 * vector_sort: Sorterar innehåll lagrat i dynamisk vektor i valbar ordning.
 *****/
void vector_sort(struct vector* self, const enum sort_direction sort_direction)
{
    for (register size_t* i = self->data; i < self->data + self->size - 1; ++i)
    {
        for (register size_t* j = i + 1; j < self->data + self->size; ++j)
        {
            if ((sort_direction == SORT_DIRECTION_ASCENDING && *i > *j) ||
                (sort_direction == SORT_DIRECTION_DESCENDING && *i < *j))
            {
                const size_t temp = *i;
                *i = *j;
                *j = temp;
            }
        }
    }
    return;
}

/*****
 * vector_print: Skriver ut innehåll lagrat i en vektor till en valbar utenhet.
 *****/
void vector_print(const struct vector* self, FILE* stream)
{
    if (!stream) stream = stdout;
    fprintf(stream, "-----\n");
    for (register size_t* i = self->data; i < self->data + self->size; ++i)
        fprintf(stream, "%zu\n", *i);
    fprintf(stream, "-----\n\n");
    return;
}

```