CPROUC/2021-2022

Jesse op den Brouw

CPROUC

Bestanden & compilatieproces

DE HAAGSE HOGESCHOOL

Bestanden

- Op een computersysteem dat met bestanden werkt (Windows, Linux, mac-OS-X, is het met C mogelijk om met deze bestanden te werken.

- Mogelijkheden:
  - Bestanden openen voor lezen of schrijven,
  - Data uit bestanden lezen
  - Data naar bestanden schrijven
  - Bestanden sluiten

Bestanden

- Op Windows wordt gebruik gemaakt van *schijfnamen*.

- De eerste vaste schijf (SSD, HDD) heet `C`, aangegeven met `C:`
- De tweede vaste schijf heet `D`, aangegeven met `D:`
- Enzovoorts…

- Als de schijf meerdere *partities* heeft, wordt ook een schijfnaam gebruikt.

Bestanden

- Een modern *bestandssysteem* gebruikt mappen om de bestanden te organiseren.

- Voorbeeld:

  `C: → Windows → system32 → cmd.exe`

- In Windows wordt dit aangegeven met:

  `C:\Windows\system32\cmd.exe`

Bestanden

- Op mac-OS-X én Linux werkt het anders. Er worden geen schijfnamen gebruikt.

- Eén schijf (of partitie) wordt gebruikt als *root* (het begin) aangegeven met een enkele *forward slash* (`/`).

- Mappen worden gescheiden met een forward slash:

```
/home/piet/huiswerk/opgave5.12/main.c
```

Bestanden

- Om bij een bestand te komen moeten we de *padnaam* opgeven.
- Dit kan absoluut of relatief.
- Absoluut:

  `C:\Windows\system32\cmd.exe`

  `/home/piet/huiswerk/opgave5.12/main.c`

- Relatief ten opzichte van de *huidige map* waar je in werkt:

`output.txt`	bestand in huidige map
`opgave5.12/main.c`	duik eerst in map opgave5.12

Bestanden

- Speciale mapnaam: `..` (twee punten)
- Dit is een referentie naar de bovenliggende map (de *parent*)
- Dus als `C:\Windows\system32` en map is, dan is de `..` (in C:\Windows\system32) `C:\Windows`
- Daarmee kan je relatief ten opzichte van de huidige map werken: `..\opgave8.17\main.c`

Tekstbestanden

- Een tekstbestand bevat in ASCII-gecodeerde data (`char`'s).
- Tekstbestanden kunnen eenvoudig bewerkt worden (b.v. met Notepad. )
- Tekstbestanden kunnen ook met een C programma eenvoudig aangemaakt, beschreven en uitgelezen worden.

Bestanden

- C kent een aantal functies met betrekking tot bestanden:

```
FILE *fopen(filename, mode);
```

- `fopen` geeft pointer naar een *file handle* terug, of `NULL` als het bestand niet benaderd kan worden. Voorbeeld:

Dubbele backslash nodig als scheider tussen mappen

```
FILE *fp = fopen("c:\\temp\\textfile.txt", "r");
```

opent een bestand voor lezen, andere mogelijkheden voor `mode`: `"w"`, voor schrijven (huidige inhoud wordt gewist), `"a"` voor toevoegen (append).

Bestanden

- Lezen uit een bestand:

```
fscanf(fp, "%d", &i);
```

- Schrijven naar een bestand:

```
fprintf(fp, "Variabele i is %d\n", i);
```

- Sluiten van een bestand:

```
fclose(fp);
```

Tekstbestand schrijven

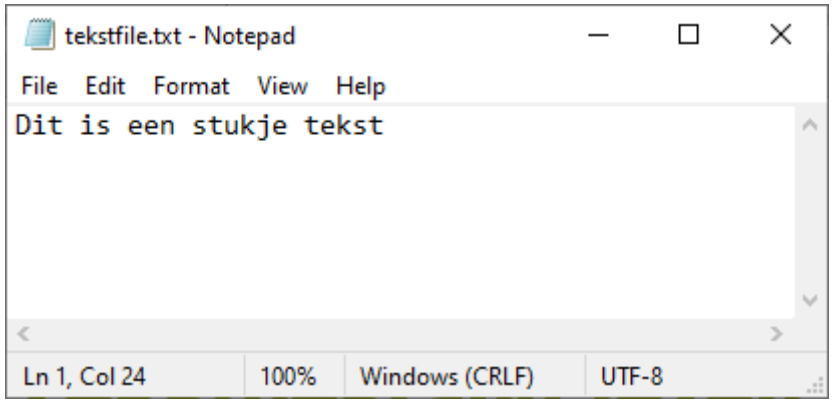
```c
#include <stdio.h>
#pragma warning(disable : 4996)

int main(void) {
    FILE *fp;
    fp = fopen("c:\\temp\\tekstfile.txt", "w");
    if (fp == NULL) {
        printf("Kan bestand niet openen!");
        exit(-1);
    }
    fprintf(fp, "Dit is een stukje tekst\n");
    fclose(fp);
}
```

Nodig voor bestandsfuncties

Keer direct terug naar het OS

Tekstbestand lezen

```c
#include <stdio.h>
#pragma warning(disable : 4996)
int main(void) {
    FILE *fp;
    char buffer[100];
    fp = fopen("c:\\temp\\tekstfile.txt", "r");
    if (fp == NULL) {
        printf("Kan bestand niet openen!");
        exit(-1);
    }
    fscanf(fp, "%99[^\n]s", buffer);
    printf("%s", buffer);
    fclose(fp);
}
```



Lees in zolang geen \n gelezen wordt, maximaal 99 karakters. Leest ook spaties.

DE HAAGSE HOGESCHOOL

Bestanden openen vanaf de command line

```c
#include <stdio.h>
#pragma warning(disable : 4996)
int main(int argc, char *argv[]) {
    FILE *fp;
    char buffer[100];
    fp = fopen(argv[1], "r");
    if (fp == NULL) {
        printf("Kan bestand niet openen!");
        exit(-1);
    }
    fscanf(fp, "%99[^\n]s", buffer);
    printf("%s", buffer);
    fclose(fp);
}
```

Argument aan programma. Test eerst of het argument bestaat (`argc == 2`)

Standaard geopende bestanden

- In een C-”omgeving” zijn altijd drie bestanden standaard geopend.

  `stdin`: de standaard invoer, het toetsenbord
  `stdout`: de standaard uitvoer, het beeldscherm
  `stderr`: de standaard error-uitvoer, het beeldscherm

- Je kan dus altijd lezen/schrijven van de standaard bestanden, tenzij je ze sluit met `fclose`.

```
fprintf(stderr, "Fout gedetecteerd!\n");
```

Compilatieproces

- Voordat een uitvoerbaar bestand is gemaakt, moet de C-compiler flink wat werk doen.
- Eigenlijk is de C-compiler meer dan één programma.
- Het is een verzameling programma's.
- Dat wordt een *toolchain* genoemd.
- We bespreken een veel gebruikte compiler: de GNU C-compiler.
- Deze wordt o.a. gebruikt bij de Arduino Uno en in Code::Blocks.

Compilatieproces

- Om een enkel C-bestand te compileren, gebruiken we

```
gcc -o prog.exe prog.c
```

- Er is nu een uitvoerbaar programma dat gestart kan worden.

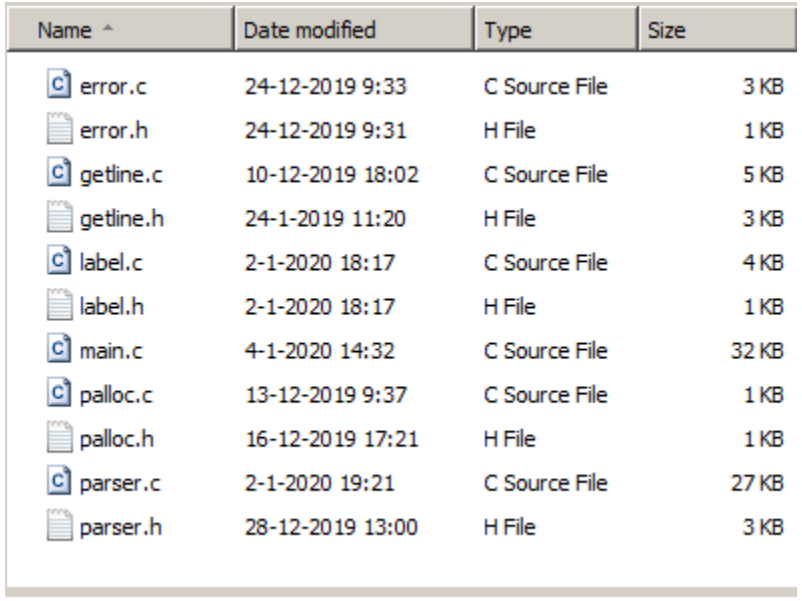
- Normaal gesproken wordt alles geregeld door de Integrated Development Environment (Code::Blocks, Arduino IDE)

Compilatieproces

- De *preprocessor* verwerkt het C-bestand als eerste. De preprocessor verwerkt onder andere `#include` en `#define`. Het resultaat is een tijdelijk bestand.
- Dit tijdelijke bestand wordt door de *C-compiler* omgezet in assembler-instructies (instructies voor de processor).
- De *assembler* genereert een bestand met uitvoerbare bitpatronen (voor de processor), maar laat *onbekende referenties* (bv. voor `printf` en `scanf`) intact (want die zijn nog niet bekend).
- De *linker* vult voor de onbekende referenties met behulp van *bibliotheken* de juiste adressen (startadressen van functies en globale variabelen) in en genereert een uitvoerbare programma.

Compilatieproces

- In een groot project wordt code meestal over meerdere .c- en .h-bestanden verspreid.
- In de .h-bestanden staan declaraties van structures/arrays en prototypes van functies.
- De “echte” code staat in de .c-bestanden
- Een groot aantal bekende functies staat in zogenoemde *bibliotheken* (zoals `strlen`, `printf`, `sin`). Die zijn al gemaakt.



Name	Date modified	Type	Size
error.c	24-12-2019 9:33	C Source File	3 KB
error.h	24-12-2019 9:31	H File	1 KB
getline.c	10-12-2019 18:02	C Source File	5 KB
getline.h	24-1-2019 11:20	H File	3 KB
label.c	2-1-2020 18:17	C Source File	4 KB
label.h	2-1-2020 18:17	H File	1 KB
main.c	4-1-2020 14:32	C Source File	32 KB
palloc.c	13-12-2019 9:37	C Source File	1 KB
palloc.h	16-12-2019 17:21	H File	1 KB
parser.c	2-1-2020 19:21	C Source File	27 KB
parser.h	28-12-2019 13:00	H File	3 KB

Compilatieproces

- We kunnen een uitvoerbaar programma maken met:

```
gcc -o prog.exe parser.c label.c error.c getline.c palloc.c
```

- De standaard C-bibliotheek hoeft niet opgegeven te worden. Dat wordt automatisch door de toolchain gedaan.

- De wiskundige bibliotheek (`sin`, `cos`, `tan`) moet (soms) wel expliciet meegelinkt worden. Veel IDE's doen dat echter automatisch.

En nu verder

- Dit is het einde van een korte introductie in de programmeertaal C.
- We gaan verder met de Arduino Uno.
- Dit is een kleine processor zonder toetsenbord en beeldscherm en met een beperkt geheugen.
- Dit betekent dat bv. scanf en printf niet gebruikt kunnen worden (en bestanden bestaan ook niet).
- Toch kan er wel invoer en uitvoer plaatsvinden. Dat zie je in de volgende lessen.

let’s change