

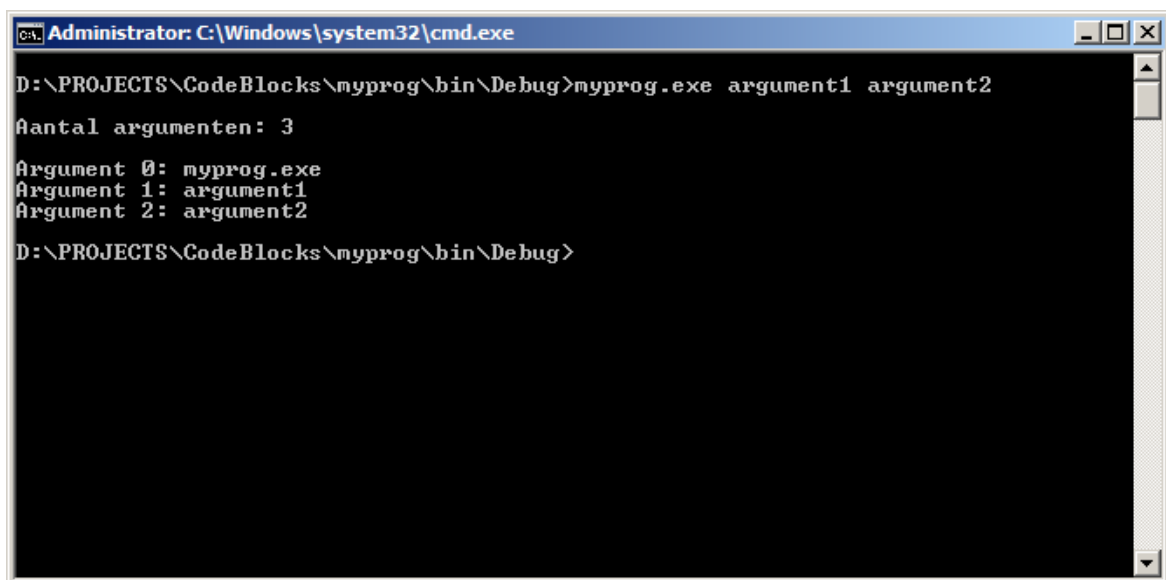
Command line argumenten en bestandsafhandeling in C

J.E.J. op den Brouw*

23 november 2017

1 Command line argumenten in C

In besturingssystemen die C ondersteunen zoals Windows, Linux en Mac OS X, is het mogelijk om een C-programma *command line argumenten* mee te geven. In dit document doen we dat in een zogenoemde DOS-box onder Windows. In figuur 1 is te zien dat het programma `myprog.exe` gestart wordt met de argumenten `argument1` en `argument2`. Het programma drukt eenvoudigweg alle argumenten op het beeldscherm af. Te zien is dat ook de programmanaam als argument wordt meegegeven.



```
Administrator: C:\Windows\system32\cmd.exe
D:\PROJECTS\CodeBlocks\myprog\bin\Debug>myprog.exe argument1 argument2
Aantal argumenten: 3
Argument 0: myprog.exe
Argument 1: argument1
Argument 2: argument2
D:\PROJECTS\CodeBlocks\myprog\bin\Debug>
```

Figuur 1: Voorbeeld van een commando met argumenten.

Elk C-programma krijgt per definitie de twee parameters `argc` en `argv` mee, die aan `main()` worden meegegeven. Dit is te zien in listing 1.

*De Haagse Hogeschool, J.E.J.opdenBrouw@hhs.nl

```

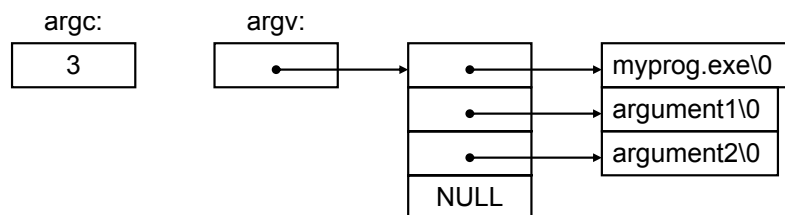
1 int main(int argc, char *argv[]) {
2
3     /* rest of the code */
4
5     return 0;
6 }

```

Listing 1: Declaratie van de command line parameters.

De integer `argc` (**argument count**) geeft aan hoeveel parameters aan het C-programma zijn meegegeven. De pointer `*argv[]` (**argument vector**) is een pointer naar een lijst van pointers naar strings. Elke string bevat een argument. Per definitie wijst `argv[0]` naar een string waarin de programmanaam vermeld staat. Dat houdt in dat `argc` dus minstens 1 is. Er zijn dan geen optionele argumenten meegegeven.

In het voorbeeldprogramma is `argc` dus 3 en zijn `argv[0]`, `argv[1]` en `argv[2]` pointers naar respectievelijk "myprog.exe", "argument1" en "argument2". In figuur 2 is een uitbeelding van de variabelen `argc` en `argv` te zien. De strings worden, zoals gebruikelijk in C, afgesloten met een `\0`-karakter. De C-standaard schrijft voor dat de lijst van strings wordt afgesloten met een `null`-adres.



Figuur 2: Uitbeelding van de variabelen `argc` en `argv`.

Het programma `myprog.exe` is te zien in listing 2. Het programma drukt eerst de variabele `argc` af. Met behulp van een `for`-lus worden de argumenten één voor één afgedrukt. Merk op dat `argv[i]` een pointer is naar het i^e argument. We kunnen `argv[i]` dus direct gebruiken voor het afdrukken van de bijbehorende string.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char *argv[]) {
5
6     int i;
7
8     printf("\nAantal argumenten: %d\n\n", argc);
9     for (i = 0; i < argc; i++) {
10         printf("Argument %d: %s\n", i, argv[i]);
11     }
12     return 0;
13 }

```

Listing 2: Het programma `myprog.exe`.

2 Bestandsafhandeling in C

In C is het mogelijk om met bestanden op de harde schijf (of USB-stick of SD-card) te werken. We spreken dan over File I/O.

In listing 3 is de code te zien om een bestand te schrijven. We zullen er stap voor stap doorheen lopen. In regel 6 wordt de pointer `outfile` naar een object `FILE` gedeclareerd. Met behulp van de pointer kunnen we onder andere lezen van een bestand en schrijven naar een bestand. In regel 8 wordt het bestand `C:\temp\testbestand.txt` geopend voor schrijven. Dat is te zien aan de tweede argument `"w"`. Merk op dat in de bestandsnaam gebruik is gemaakt van een dubbele backslash omdat een backslash door C wordt gebruikt voor afdrukkbare karakters zoals `\n`. De functie `fopen` geeft een adres terug die wordt toegekend aan de pointer `outfile`. Als om één of andere reden het bestand niet geopend kan worden, geeft de functie `fopen` een `null`-adres terug. We moeten hier natuurlijk op testen. Dat is te zien in regel 10. We drukken dan een verklarende tekst af (regel 11). Als het bestand voor schrijven geopend kan worden, dan schrijven we een regel naar dat bestand. Dat is te zien regel 13. De functie `fprintf` lijkt veel op de bekende `printf`-functie maar heeft één extra parameter, namelijk de pointer `outfile`. In regel 14 wordt het bestand met behulp van de functie `fclose` gesloten.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     FILE *outfile;
7
8     outfile = fopen("C:\\temp\\testbestand.txt", "w");
9
10    if (outfile == NULL) {
11        printf("Bestand kan niet geopend worden.\n");
12    } else {
13        fprintf(outfile, "Een stukje tekst\n");
14        fclose(outfile);
15    }
16
17    return 0;
18 }
```

Listing 3: Voorbeeld schrijven naar een bestand.

Listing 4 laat de code zien om een bestand te lezen. Er wordt weer gebruik gemaakt van een `FILE`-pointer, in dit geval de pointer `infile`. Het genoemde bestand wordt weer geopend met de functie `fopen` maar ditmaal met als tweede argument `"r"` wat staat voor lezen uit een bestand. Mocht om één of andere reden het bestand niet geopend kunnen worden, dan geeft `fopen` een `null`-adres terug. Hier testen we op in regel 11.

In de regels 14 t/m 16 worden de karakters één voor één ingelezen en afgedrukt op het beeldscherm. We zien hier een typische C-`while`-constructie. Er wordt een karakter inge-

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     FILE *infile;
7     int ch;
8
9     infile = fopen("C:\\temp\\testbestand.txt", "r");
10
11     if (infile == NULL) {
12         printf("Kan bestand niet openen.");
13     } else {
14         while ((ch = fgetc(infile)) != EOF) {
15             printf("%c", ch);
16         }
17     }
18
19     fclose(infile);
20
21     return 0;
22 }

```

Listing 4: Voorbeeld lezen uit een bestand.

lezen door middel van de functie `fgetc` en toegekend aan de variabele `ch`. Tegelijkertijd wordt getest of het ingelezen karakter gelijk is aan `EOF` wat staat voor end-of-file. Zolang dat niet het geval is wordt het karakter afgedrukt met de functie `printf`.

3 Command line argumenten en bestandsafhandeling

Het laatste voorbeeld betreft het kopiëren van een bestand. We gebruiken de command line argumenten om de bestandsnamen door te geven en bestandsafhandeling om het daadwerkelijke kopiëren te realiseren. Het programma is te vinden in listing 5.

We declareren eerst de pointers voor de twee bestanden, dit is te zien in regel 8. In regel 10 declareren we twee integers, één voor de in te lezen karakters en één voor het bijhouden hoeveel karakters er zijn gekopieerd. In de regels 14 t/m 17 testen we of het aantal opgegeven argumenten drie is. Zo niet, dan drukken we een helptekst af om de gebruiker te informeren hoe het commando moet worden gebruikt. Tevens wordt het programma afgesloten met een `return`-statement. Deze geeft de waarde `-1` terug aan het besturingssysteem, een gebruikelijke techniek in Unix-achtige omgevingen om foutmeldingen door te geven.

In de regels 20 t/m 23 wordt getest of de namen van de twee bestanden identiek zijn. Dat mag natuurlijk niet anders wordt het leesbestand overschreven. We geven een passende foutmelding en verlaten het programma middels een `return`-statement met de waarde `-2`.

```

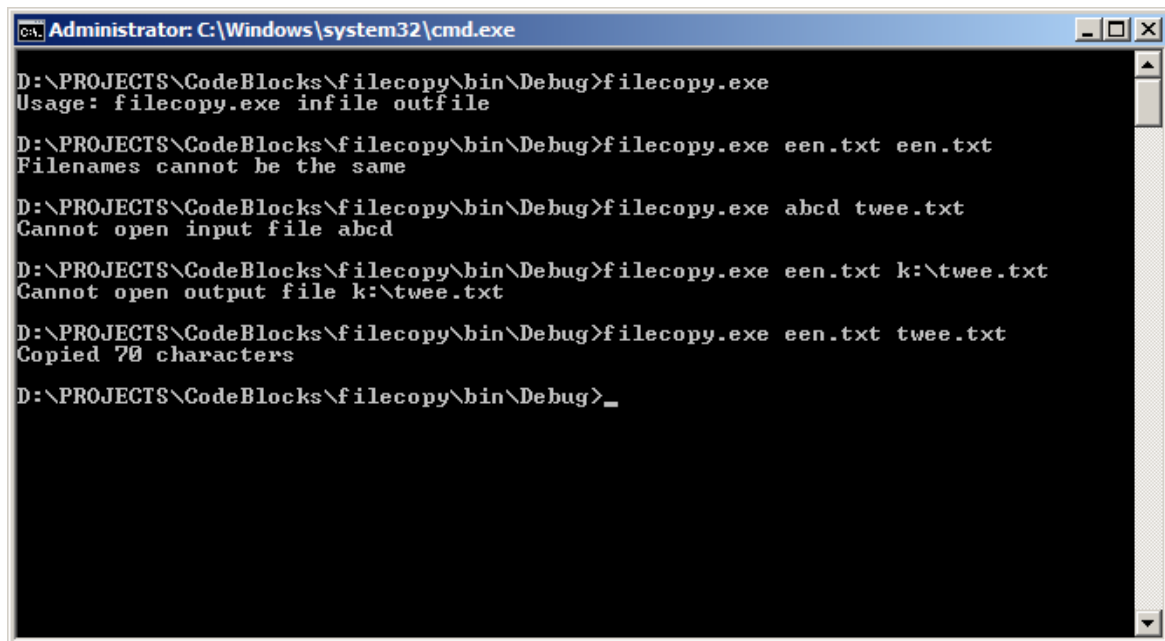
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 int main(int argc, char *argv[]) {
6
7     /* The input and output file handlers */
8     FILE *infile, *outfile;
9
10    /* Character to read and character count */
11    int ch, count = 0;
12
13    /* We need exactly 3 arguments */
14    if (argc != 3) {
15        printf("Usage: %s infile outfile\n", argv[0]);
16        return -1;
17    }
18
19    /* Test if input file and output file have the same name */
20    if (strcmp(argv[1], argv[2]) == 0) {
21        printf("Filenames cannot be the same\n");
22        return -2;
23    }
24
25    /* Open the input file, exit if error */
26    infile = fopen(argv[1], "r");
27    if (infile == NULL) {
28        printf("Cannot open input file %s\n", argv[1]);
29        return -3;
30    }
31
32    /* Open the output file, exit if error */
33    outfile = fopen(argv[2], "w");
34    if (outfile == NULL) {
35        printf("Cannot open output file %s\n", argv[2]);
36        fclose(infile);
37        return -4;
38    }
39
40    /* Copy characters from input file to output file */
41    while ((ch = fgetc(infile)) != EOF) {
42        count++;
43        fprintf(outfile, "%c", ch);
44    }
45
46    printf("Copied %d characters\n", count);
47
48    fclose(infile);
49    fclose(outfile);
50
51    return 0;
52 }

```

Listing 5: Programma om een bestand te kopiëren.

In de regels 26 t/m 30 wordt het leesbestand geopend. We hebben dit al eerder besproken. In de regels 33 t/m 38 wordt het schrijfbestand geopend. Ook dit is al eerder besproken. Het daadwerkelijke kopiëren gebeurt in de regels 41 t/m 44. Tevens wordt bijgehouden hoeveel karakters er zijn gekopieerd. Dit aantal wordt afgedrukt in regel 46.

We ronden het programma af door het leesbestand en schrijfbestand af te sluiten. Een aantal voorbeelden van het gebruik van het programma is te zien in figuur 3.

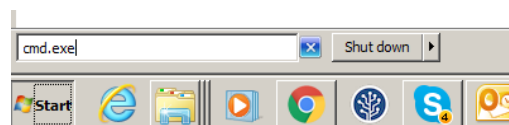


```
Administrator: C:\Windows\system32\cmd.exe
D:\PROJECTS\CodeBlocks\filecopy\bin\Debug>filecopy.exe
Usage: filecopy.exe infile outfile
D:\PROJECTS\CodeBlocks\filecopy\bin\Debug>filecopy.exe een.txt een.txt
Filenames cannot be the same
D:\PROJECTS\CodeBlocks\filecopy\bin\Debug>filecopy.exe abcd twee.txt
Cannot open input file abcd
D:\PROJECTS\CodeBlocks\filecopy\bin\Debug>filecopy.exe een.txt k:\twee.txt
Cannot open output file k:\twee.txt
D:\PROJECTS\CodeBlocks\filecopy\bin\Debug>filecopy.exe een.txt twee.txt
Copied 70 characters
D:\PROJECTS\CodeBlocks\filecopy\bin\Debug>_
```

Figuur 3: Voorbeelden van het kopieerprogramma.

4 Openen van een DOS-box op Windows

Een DOS-box kan je openen via de Start-knop (in het voorbeeld onder Windows 7). Klik op Start en vul in het kader de tekst `cmd.exe` in. Druk op de enter-toets en een DOS-box wordt geopend. Zie figuur 4.



Figuur 4: Openen van een DOS-box.

Vervolgens kun je navigeren naar de *executable* (uitvoerbaar programma) middels het commando `cd` (change directory). In het voorbeeld van figuur 3 doe je dat met het commando's :

D: (wisselen van schijf)

`cd \PROJECTS\CodeBlocks\filecopy\bin\Debug` (navigeren naar map)