



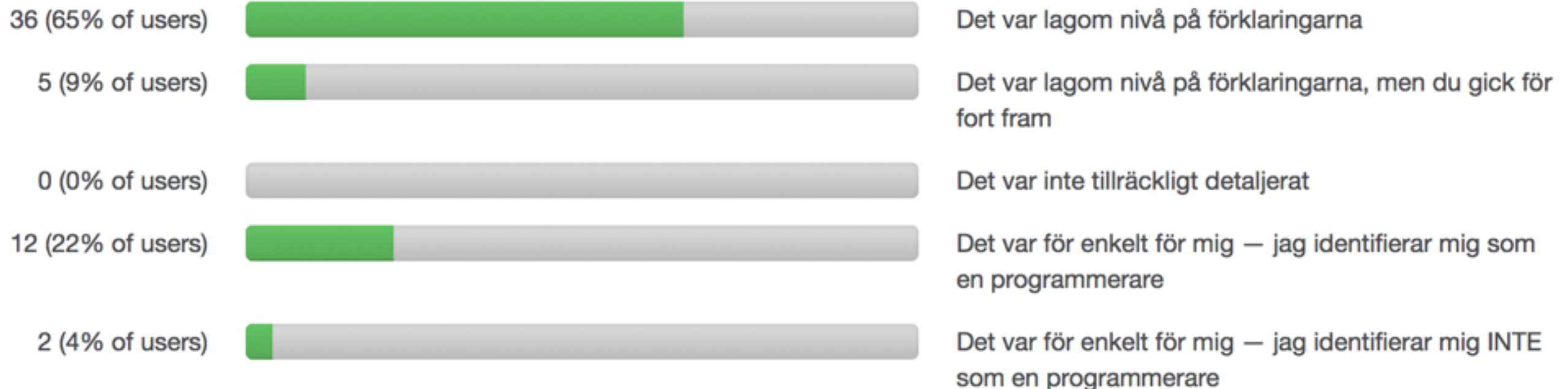
**KEEP  
CALM  
AND  
LOVE  
PROGRAMMING**

# Undersökning på Piazza

---

## Nivå på föreläsning 2 is now closed

A total of 55 vote(s) in 53 hours



# Föreläsning 3

---

Elias Castegren

*Introduktion till standard-I/O och  
grundläggande stränghantering*



# Hur var det i Haskell?

---

- Char är en egen typ

`'a' :: Char`

`'a' + 2` — *kompilerar ej*

# Hur var det i Haskell?

---

- Char är en egen typ

`'a' :: Char`

`'a' + 2` — *kompilerar ej*

- `type String = [Char]`

`"Hello" == ['H', 'e', 'l', 'l', 'o']`

`'F' : "oo" ++ "Bar!" == "FooBar!"`

# Hur var det i Haskell?

---

- Char är en egen typ

`'a' :: Char`

`'a' + 2` — *kompilerar ej*

- `type String = [Char]`

`"Hello" == ['H', 'e', 'l', 'l', 'o']`

`'F' : "oo" ++ "Bar!" == "FooBar!"`

- Kan du listor kan du strängar! (mer eller mindre)

# Tecken i C

---

- **char** — ett heltal (på *normalt* en byte)
- Ett tecken är en **char** med motsvarande ASCII-värde

```
char c1 = 'a';
```

```
char c2 = 97;           == c1
```

```
char c3 = 'a' + 2;     == 'c'
```

```
char c4 = '2' + '2';   == 50 + 50
```

# Tecken i C

---

- **char** — ett heltal (på *normalt* en byte)
- Ett tecken är en **char** med motsvarande ASCII-värde

```
char c1 = 'a';
```

```
char c2 = 97;      == c1
```

```
char c3 = 'a' + 2; == 'c'
```

```
char c4 = '2' + '2'; == 100
```



# Tecken i C

---

- **char** — ett heltal (på *normalt* en byte)
- Ett tecken är en **char** med motsvarande ASCII-värde

```
char c1 = 'a';
```

```
char c2 = 97;           == c1
```

```
char c3 = 'a' + 2;      == 'c'
```

```
char c4 = '2' + '2';    == 'd'
```

# Tecken i C

---

- **char** — ett heltal (på *normalt* en byte)
- Ett tecken är en **char** med motsvarande ASCII-värde

```
char c1 = 'a';
```

```
char c2 = 97;           == c1
```

```
char c3 = 'a' + 2;     == 'c'
```

```
char c4 = '2' + '2';   == 'd'
```

- Obs! 'a' och 97 är alltså ekvivalenta

'a' är *syntaktiskt socker* för 97

# Tecken i C

---

- Livekodning: `ascii.c`

# Strängar i C

---

Strängar i C är  
Bara arrayer av chars  
som slutar med '\0'



# Strängar i C

---

- C har ingen strängtyp!

`char*` — en *pekare\** till en följd av chars i minnet (*\*se föreläsning 5*)

`char[]` — en array av chars

I stora stycken är typerna ovan ekvivalenta

# Strängar i C

---

- C har ingen strängtyp!

`char*` — en pekare till en följd av chars i minnet

`char[]` — en array av chars

I stora stycken är typerna ovan ekvivalenta

- Kan du arrayer kan du strängar! (mer eller mindre)

`char s[] = "Hello"` är **precis** samma sak som

`char s[] = {'H', 'e', 'l', 'l', 'o', '\0'}` vilket är **precis** samma sak som

`char s[] = {72, 101, 108, 108, 111, 0}`

# Strängar i C

---

- C har ingen strängtyp!

`char*` — en pekare till en följd av chars i minnet

`char[]` — en array av chars

I stora stycken är typerna ovan ekvivalenta

- Kan du arrayer kan du strängar! (mer eller mindre)

`char s[] = "Hello"` är **precis** samma sak som

`char s[] = {'H', 'e', 'l', 'l', 'o', '\0'}` vilket är **precis** samma sak som

`char s[] = {72, 101, 108, 108, 111, 0}`

s =	'H'	'e'	'l'	'l'	'o'	'\0'
-----	-----	-----	-----	-----	-----	------



# Strängar i C

---

- C har ingen strängtyp!

`char*` — en pekare till en följd av chars i minnet

`char[]` — en array av chars

I stora stycken är typerna ovan ekvivalenta

- Kan du arrayer kan du strängar! (mer eller mindre)

`char s[] = "Hello"` är **precis** samma sak som

`char s[] = {'H', 'e', 'l', 'l', 'o', '\0'}` vilket är **precis** samma sak som

`char s[] = {72, 101, 108, 108, 111, 0}`

s = 

'H'	'e'	'l'	'l'	'o'	'\0'
-----	-----	-----	-----	-----	------

*Hello är en array av **sex** chars!*

# Vanliga misstag 1

- Strängjämförelse med `==` avser *identitet*, inte *ekvivalens*

```
#include <stdio.h>

int main (void)
{
    char password[] = "abc123";
    char entered[128];

    puts("Please enter the secret code:");
    scanf("%s", entered); // read input

    if (entered == password)
    {
        puts("You are logged in!");
    }
    else
    {
        puts("Incorrect password!");
    }

    return 0;
}
```

# Vanliga misstag 1: Strängjämförelse

---

- Livekodning: `password.c`

# Vanliga misstag 2

- Aliasering!

```
#include <stdio.h>

int main (void)
{
    char buffer [128];
    char *first;
    char *last;

    puts("What is your first name?");
    scanf("%s", buffer);
    first = buffer;

    puts("What is your last name?");
    scanf("%s", buffer);
    last = buffer;

    printf("Hello %s %s!\n", first, last);
    return 0;
}
```

# Vanliga misstag 2: Aliasering

---

- Livekodning: `greeting.c`

# Vanliga misstag 3

---

- ...som slutar med '\0'!

```
#include <stdio.h>

int main (void)
{
    char *s = "Hello";
    char t[5];

    copy(t, s);

    puts(t);

    return 0;
}
```

# Vanliga misstag 3: ...som slutar med '\0'!

---

- Livekodning: `length.c`

# Standardbiblioteket `string.h`

- Inkluderas med `#include <string.h>`

Funktion	Beskrivning
<code>strlen(s)</code>	längden av <code>s</code> (utan <code>'\0'</code> -tecknet!)
<code>strncpy(s, t, n)</code>	kopiera <code>t</code> till <code>s</code> (upp till <code>n</code> tecken)
<code>strncmp(s, t, n)</code>	jämför <code>s</code> och <code>t</code> (upp till <code>n</code> tecken)
<code>== 0</code>	<code>s</code> och <code>t</code> är samma sträng
<code>&gt; 0</code>	<code>s</code> kommer efter <code>t</code> i bokstavsordning
<code>&lt; 0</code>	<code>s</code> kommer före <code>t</code> i bokstavsordning
<code>strncat(s, t, n)</code>	Lägg <code>t</code> följt av <code>s</code> i <code>t</code>



# Funktioner i `stdlib.h` (togs även upp tidigare)

---

- Inkluderas med `#include <stdlib.h>`

`atoi(s)` — konvertera strängrepresentation av tal till motsvarande heltal

`atol(s)` — med `long` istf `int` som returtyp

Exempel:

`atoi("123abc") == 123` (som `int`)

# OBS!

---

- Alla funktioner är optimistiska

Förutsätter från att det finns tillräckligt minne i datat som används

Förutsätter att strängar är '`\0`'-terminerade korrekt

- Använd alltid `strncpy` (`strncpy`, ...) över `strcpy` (`strcpy`, ...) etc.

# OBS!

---

- Alla funktioner är optimistiska

Förutsätter från att det finns tillräckligt minne i datat som används

Förutsätter att strängar är `'\0'`-terminerade korrekt

- Använd alltid `strncpy` (`strncpy`, ...) över `strcpy` (`strcpy`, ...) etc.
- Använd alltid funktioner från standardbiblioteken (som `string.h`) över funktioner du skriver själv

# OBS!

---

- Alla funktioner är optimistiska

Förutsätter från att det finns tillräckligt minne i datat som används

Förutsätter att strängar är `'\0'`-terminerade korrekt

- Använd alltid `strncpy` (`strncpy`, ...) över `strcpy` (`strcpy`, ...) etc.
- Använd alltid funktioner från standardbiblioteken (som `string.h`) över funktioner du skriver själv

Även om det är en bra övning att ha implementerat motsvarande funktioner själv någon gång!

```
#include <stdio.h>

int main (void)
{
    char password[] = "abc123";
    char entered[128];
    size_t entered_size; // längden på entered

    puts("Please enter the secret code:");
    scanf("%s", entered); // read input
    entered_size = strlen(entered);

    if (strncmp(entered, password, entered_size) == 0)
    {
        puts("You are logged in!");
    }
    else
    {
        puts("Incorrect password!");
    }

    return 0;
}
```

password-good.c

# Standardbiblioteket `ctype.h`

- Inkluderas med `#include <ctype.h>`

Funktion	Beskrivning
<code>isalpha(c)</code>	Är <code>c</code> en bokstav?
<code>isdigit(c)</code>	Är <code>c</code> en siffra?
<code>islower(c)</code>	Är <code>c</code> en gemen? (liten bokstav)
<code>digittoint(c)</code>	Konvertera från '2' till 2 (som int)
<code>toupper(c)</code>	Från gemen till versal ('a' till 'A')

# Efter pausen: Grundläggande I/O

---



# Alla program exekverar i ett sammanhang!

---

```
int main (void)
{
    puts("Hello world!");
    return 0;
}
```

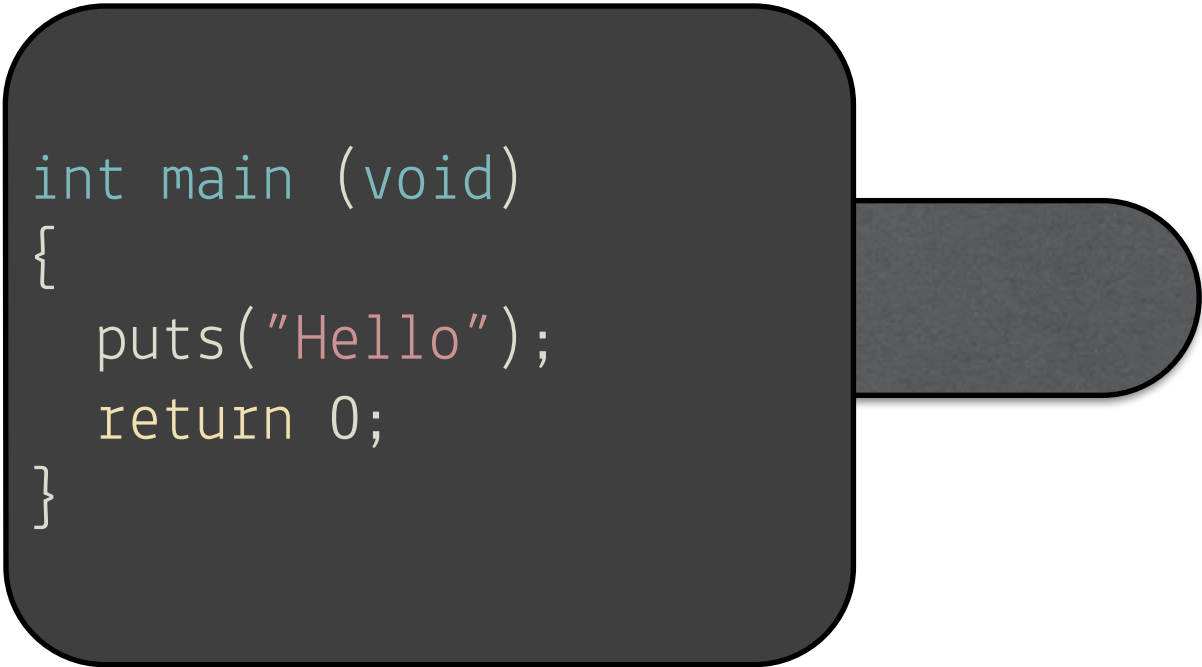
Hello world!



# Alla program exekverar i ett sammanhang!

---

- Kommunikation sker genom *strömmar* (eng. *streams*)
- Ett programs utström...
- ...är ett annat programs inström!



```
int main (void)
{
    puts("Hello");
    return 0;
}
```

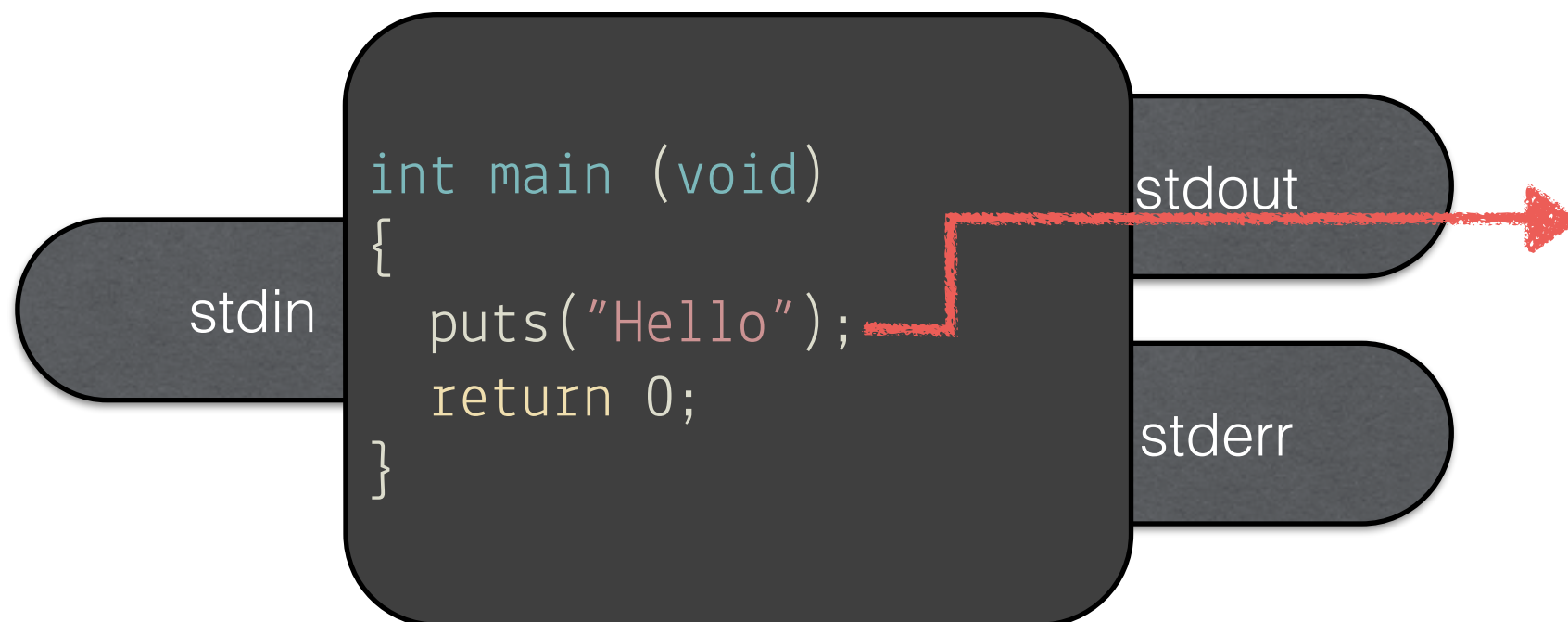
# Standardströmmarna

- Alla program har (minst) tre strömmar:

stdout — hit skrivs programmets output

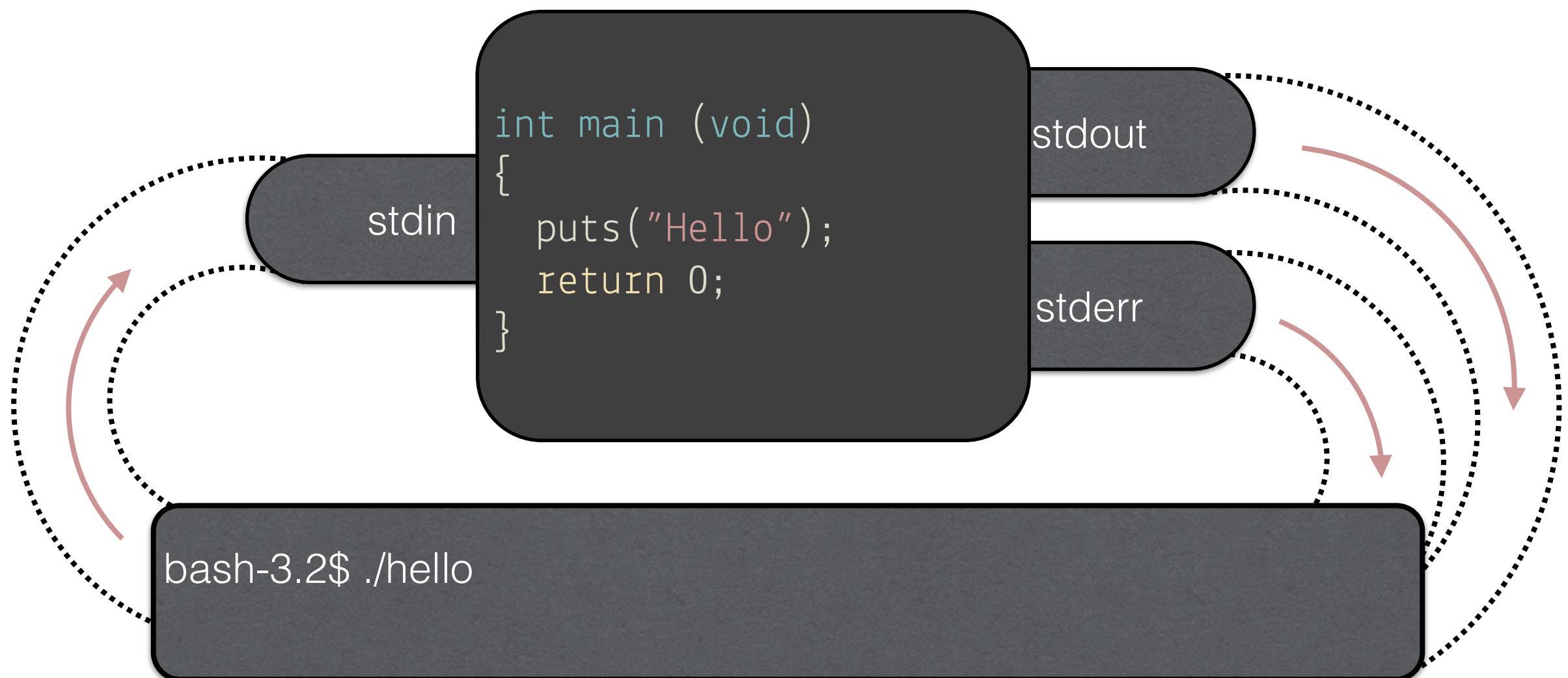
stdin — härifrån läses programmets input

stderr — hit skrivs programmets felmeddelanden



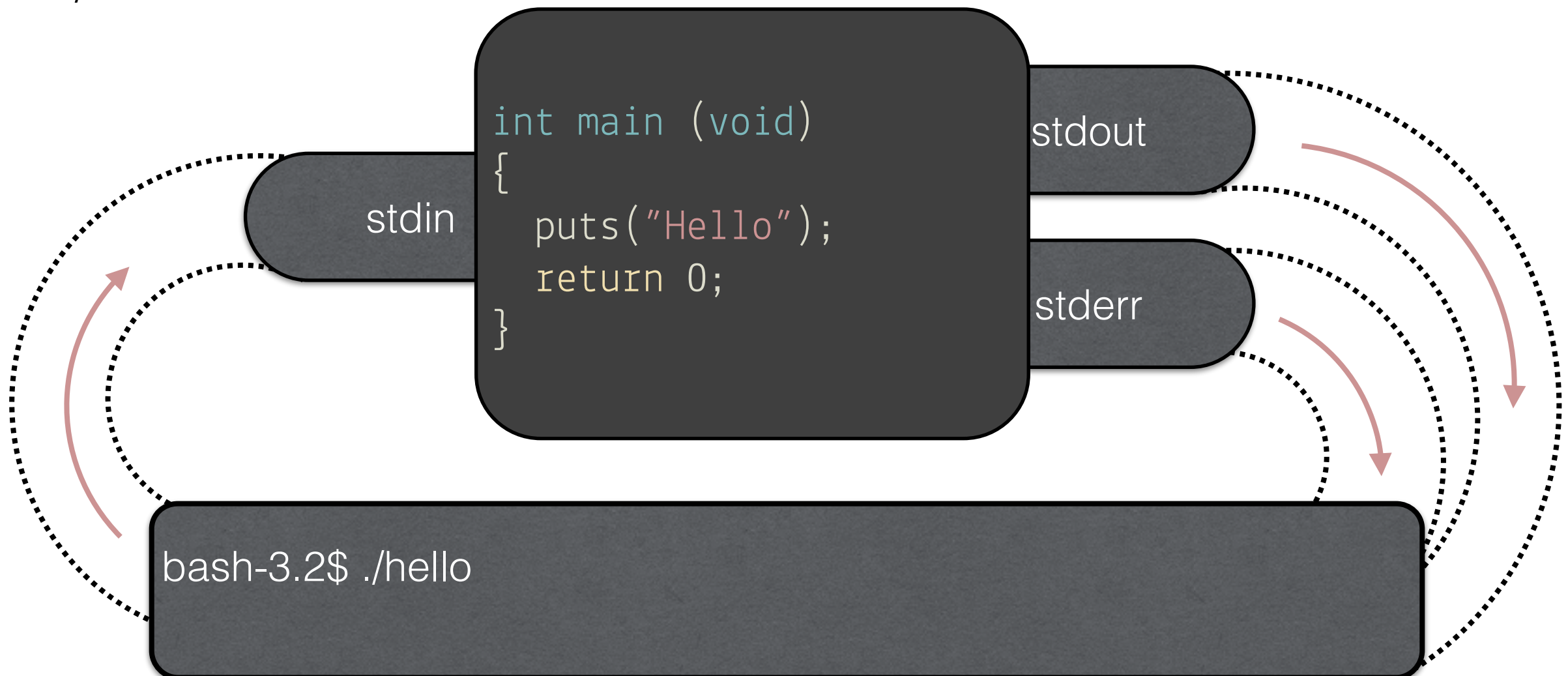
# Standardströmmarna

- När vi kör ett program som vanligt kopplar terminalen in sig i programrets strömmar



# Standardströmmarna

- Terminalen låter oss koppla ihop program!  
`./foo | ./bar` — Koppla foo:s stdout till bar:s stdin  
`./foo > out.txt` — Skriv foo:s stdout till out.txt  
`./foo < in.txt` — Kör foo och läs stdin från in.txt



# Standardbiblioteket `stdio.h`

---

- Inkluderas med `#include <stdio.h>`

Funktion	Beskrivning
<code>putchar(c)</code>	Skriv en char till stdout
<code>puts(s)</code>	Skriv en sträng till stdout
<code>printf(s, ...)</code>	Skriv en formaterad sträng till stdout

- Egentligen specialfall av mer generella funktioner:

`putchar(c) == fputc(c, stdout)`

`puts(s) == fputs(s, stdout)`

`printf(s, ...) == fprintf(stdout, s, ...)`

# Standardbiblioteket `stdio.h`

- Inkluderas med `#include <stdio.h>`

Funktion	Beskrivning
<code>getchar()</code>	Läs en char från stdin
<code>fgets(s, n, stdin)</code>	Läs en rad från stdin till s (max n chars)
<code>scanf(s, ...)</code>	Läs formaterat från stdin

- Egentligen specialfall av mer generella funktioner:

`getchar(c) == fgetc(c, stdin)`

`scanf(s, ...) == fscanf(stdin, s, ...)`

- **OBS!** Använd inte `scanf` för att läsa strängar!

# Standardbiblioteket `stdio.h`

---

- Livekodning: `megaphone.c`