Eine portable Schnittstelle zum Aufruf von C-Funktionen

M. Anton Ertl
TU Wien

Warum und Was?

- C ist die Verkehrssprache unter den Programmiersprachen
- Viele Libraries können von C aus aufgerufen werden
- C-Funktionen aufrufen (implementiert)
- Rückruf von C nach Forth (teilweise)
- Zugriff auf C-Datenstrukturen (noch nicht portabel)

Ziele

- Aufrufe portabel
- Deklarationen portabel
- Bequem für den Programmierer
- Keine Daten abschneiden
- Aber auch alle möglichen Werte übergeben

Portabilität

```
#if ... /* platform-specific */
typedef long      off_t;
#else
typedef long long off_t;
#endif
off_t lseek(int fildes, off_t offset, int whence);
```

oder auf alten Systemen:

```
off_t lseek(int fildes, long offset, int whence);
```

Welchen Stack-Effekt soll das Forth-Wort haben?

```
lseek ( n n n -- n )
lseek ( d d d -- d )
lseek ( n d n -- d )
```

Typen

C	Forth
`char, short, int, long, long long`	cell, double-cell
Zeiger	cell (Address)
`float, double, long double`	Float

Reihenfolge der Parameter

```
void sincos(double x, double *sin, double *cos);
```

```
fvariable sin
fvariable cos
1e sin cos sincos sin f@ cos f@ \ Vorwärts
cos sin 1e sincos sin f@ cos f@ \ Rückwärts
```

Vorwärts ist für den Programmierer bequemer

Lösung

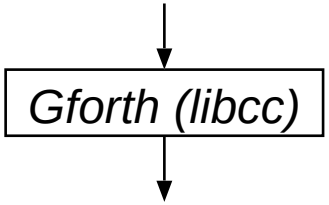
- Deklaration der Forth-Seite (portabel)
  `c-function dlseek lseek n d n -- d`

- Deklaration der C-Seite (plattform-spezifisch)
  - Verwendung der `.h`-Dateien (libcc)
  - oder als Deklaration (libffi, ffcall):
    `c-types lseek int longlong int -- longlong` oder
    `c-types lseek int long int -- long`

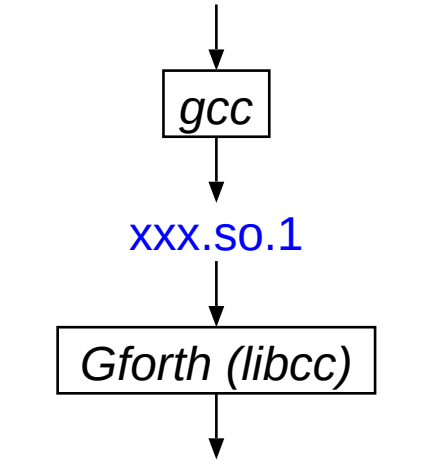
- Aufruf (portabel):
  `fd @ 0. SEEK_SET dlseek -1. d= if ...`

libcc

```
\c #include <unistd.h>
c-function dlseek lseek n d n -- d
```




```
#include <unistd.h>
void gforth_c_lseek_ndn_n(void)
{
  Cell MAYBE_UNUSED *sp = gforth_SP;
  Float MAYBE_UNUSED *fp = gforth_FP;
  sp[3]=lseek(sp[3],gforth_d2ll(sp[2],sp[1]),sp[0]);
  gforth_SP = sp+3;
}
```



Varargs

```
printf("%n %n", n1, n2);
printf("%f", r1);
```

```
c-function nn-printf printf a n n -- n
\ c-types printf ptr int int -- int
c-function r-printf printf w r -- n
\ c-types printf ptr double -- int

s\" %n %n\0" drop 23 42 nn-printf .
s\" %f\0"    drop 2.5e   r-printf .
```

Groß- und Kleinschreibung

- C beachtet den Unterschied
- Forth (im allgemeinen) nicht
- C-Namen können in Forth kollidieren
- C-Namen können mit Forth-Namen kollidieren
- Lösung: Umbenennen:

```
c-function foo1 Foo ...
c-function foo2 foo ...
```

- Alternative: Eigenen wordlist (case-sensitive)

Rückruf

```
void qsort(void *base, size_t nmemb, size_t size,
           int(*compar)(const void *, const void *));

: n-compare ( addr1 addr2 -- n )
 @ swap @ swap - ;

: sort-cells ( addr u -- )      \ idealized
  1 cells ['] n-compare qsort ;

c-function-ptr       compar w w -- n
c-function-ptr-types compar ptr ptr -- int

['] n-compare compar fptr-n-compare

: sort-cells ( addr u -- )      \ realistic
  1 cells fptr-n-compare qsort ;
```

Zukunft

- Implementation des Rückrufs
- Implementation mit libffi, ffcall
- Caching der generierten Wrapper-Funktionen
- Datenstrukturen

Zusammenfassung

- Ziel: portable Aufrufe
- Problem: Unterschiedliche C-Typen auf unterschiedlichen Plattformen
- Lösung:
  Deklaration der Forth-Seite (portabel)
  Erzeugung der C-Seite mit Hilfe von .h-Dateien
  Calls (portabel)