

## Start CLAR | C - Linux - Arduino - Raspberry

---

### Inhaltsverzeichnis

#### Start CLAR | C - Linux - Arduino - Raspberry

##### Projekt c/

Homepage und Downloads	2
Allgemeine Dokumentationen	2
Dokumentation: C-Programme und Bibliotheken	2

##### A| Projekt c/ auf PC oder PI einrichten

1. Voraussetzungen	3
2. Download	3
3. Archiv entpacken	3
4. chelp einrichten	3
Fehlerfall	4

##### B| Projekthilfe chelp einrichten

chelp kompilieren	5
Startcheck	5
Einstellungen	6
Hilfe zum Projekt c/	6

##### C| Programm infosys

Programm 'infosys' mit chelp compilieren	7
Programm infosys	7
Infos zum Keyboard	8
Infos zum Rechner	8
Infos zum Speicher	9

##### D| Ein eigenes C Programme erstellen

Hilfs-Programm 'newprg' compilieren	10
Neues Programm 'newprg' erstellen	10

##### Programme compilieren

1. Programme mit make compilieren	11
2. Programme mit chelp compilieren	11
3. Programme mit Entwicklungsumgebung geany compilieren	11
Fehler beim Compilieren	11

##### Datei- und Ordnerübersicht

##### GNU General Public License

---

---

## Projekt c/

### Homepage und Downloads

**Homepage und Downloads:** [www.projektc.at](http://www.projektc.at)

---

Projekt c/ stellt Werkzeuge und Informationen zum Entwickeln von C-Programmen mit Linux auf PC's, Raspberry und Arduino bereit.

- ▷ Alle Dateien befinden sich in Unterverzeichnissen des Projektordners c/ . Optional können weiter Ordner eingebunden werden.
- ▷ Der Name des Projektordners ist zwingend 'c/'. Zur Identifizierung des Ordners dient die Lizenzdatei c/GNU\_Licence.
- ▷ Die Lage des Projektordners c/ ist beliebig. Er kann auf internen oder externen Dateisystemen (z.B USB) eingerichtet werden.
- ▷ Es wird nichts im Linux-System installiert oder geändert!

Die Anleitung beschreibt das Einrichten von Projekt c/ auf [Raspberry Pi](#) und [Linux PC](#).

### Allgemeine Dokumentationen

Die allgemeinen Dokumentationen findet man unter [c/1\\_Dokus](#) oder im Internet:

Vorwort:	<a href="http://www.projektc.at/programs/c/clar_vorwort.pdf">www.projektc.at/programs/c/clar_vorwort.pdf</a>	<a href="#">c/1_Dokus/clar_vorwort.pdf</a>
Projekt c/ einrichten. Die ersten Schritte:	<a href="http://www.projektc.at/programs/c/clar_start.pdf">www.projektc.at/programs/c/clar_start.pdf</a>	<a href="#">c/1_Dokus/clar_start.pdf</a>
Projekthilfe und Projektmanager:	<a href="http://www.projektc.at/programs/c/clar_chelp.pdf">www.projektc.at/programs/c/clar_chelp.pdf</a>	<a href="#">c/1_Dokus/clar_chelp.pdf</a>
Ein neues C Programm erstellen:	<a href="http://www.projektc.at/programs/c/clar_projekt.pdf">www.projektc.at/programs/c/clar_projekt.pdf</a>	<a href="#">c/1_Dokus/clar_projekt.pdf</a>
Basisobjekte ohne Terminal In/Ouput:	<a href="http://www.projektc.at/programs/c/clar_objekte1.pdf">www.projektc.at/programs/c/clar_objekte1.pdf</a>	<a href="#">c/1_Dokus/clar_objekte1.pdf</a>
Terminalsteuerung Box-Objekte für In/Ouput:	<a href="http://www.projektc.at/programs/c/clar_objekte2.pdf">www.projektc.at/programs/c/clar_objekte2.pdf</a>	<a href="#">c/1_Dokus/clar_objekte2.pdf</a>
Datenspeicherung:	<a href="http://www.projektc.at/programs/c/clar_objekte3.pdf">www.projektc.at/programs/c/clar_objekte3.pdf</a>	<a href="#">c/1_Dokus/clar_objekte3.pdf</a>

### Dokumentation: C-Programme und Bibliotheken

- ▷ Die ersten Infos zu den C-Programmen oder Bibliotheken findet man in der Hilfedatei '1\_read.me' im jeweiligen Programmordner.
- ▷ Für aufwendige Programme gibt es Beschreibungen im Format \*.odt oder \*.pdf im Ordner ProgName/bin/\_ProgName/
- ▷ Der Programmcode ist in den C-Headern der Programme oder Bibliotheken dokumentiert.
- ▷ Hilfe zu den fertigen Programmen liefert immer die Programm-Startoption '-h'.

Einstiege:

Programm chelp	<a href="#">Verwaltung von Projekt c/. Menügesteuerter Zugriff auf alle Dokus</a>
Projekt c/ Einstieg und Übersicht	<a href="#">c/1_read.me</a>
Header/Dokus für Bibliotheksfunktionen	<a href="#">c/lib/1_read.me</a>
Testprogramme für Bibliotheksfunktionen	<a href="#">c/libtest/1_read.me</a>

---

## A) Projekt c/ auf PC oder PI einrichten

### 1. Voraussetzungen

- ▷ Linux System
- ▷ Datenträger mit symbolischen Links!
- ▷ Der Name des Projektordners ist zwingend 'c'
- ▷ Die Lage des Ordners im Dateisystem ist beliebig!
- ▷ Im Projektordner wird die Lizenzdatei `c/GNU_Licence` abgelegt
- ▷ Option: Startlinks für die Programme im Homeverzeichnis `~/bin`
- ▷ X-Software: `geany` Entwicklungsumgebung, Textverarbeitung mit Option `goto` Zeilennummer und ein Starter für Standard-Programme des Benutzers z.B. `xdg-open`

### 2. Download

Homepage:	<a href="http://www.projektc.at/programs/c">www.projektc.at/programs/c</a>	Projekt /c Werkstatt
Download 1:	<a href="#">c.tar.gz</a>	Projektarchiv in <code>~/Downloads</code>
Download 2:	<a href="#">shasum.txt</a>	Prüfsumme mit Anleitung in <code>~/Downloads</code>

### 3. Archiv entpacken

- ▷ Beispiel: Zielordner `/home/user/test/c/`  
Im Terminal ausführen:
 

```
cd ~ # In das Homeverzeichnis wechseln
mkdir -pv ~/test/c # Ordner und Projektordner c anlegen
cd test/c # In den Ordner c wechseln
mv ~/Downloads/c.tar.gz ./ # Archiv in den Ordner test/c kopieren
cat ~/Downloads/shasum.txt # Prüfsumme und Befehl auslesen
Datum : Mo 22 Sep 2025 22:51:12 CEST
Befehl: shasum -a 256 c.tar.gz
42b0ef5b891dacc5d6833d0cd66ae074413a99745812b2627d22954d32d4f3e5 c.tar.gz
shasum -a 256 c.tar.gz # Prüfsumme des Archivs ermitteln
42b0ef5b891dacc5d6833d0cd66ae074413a99745812b2627d22954d32d4f3e5 c.tar.gz
tar -xzf c.tar.gz -C ~/test # Archiv in Ordner Test entpacken
tree | less # Ordner c prüfen
```

```

├── 1_Dokus
│   ├── clar_chelp.odt
│   ├── clar_chelp.pdf
│   └── clar_objekte1.odt
├── bin
│   ├── 1_read.me
│   ├── archivfoto
│   │   ├── archivfoto.c
│   │   ├── archivfoto.h
│   │   └── bin
│   │       └── _archivfoto
│   └── 1_read.me
└── .

```

### 4. chelp einrichten

Mit `cd /home/user/test/c` in den Projektordner wechseln:

- ▷ Befehl `make`
- ▷ Fehlerfall: `make: makefile: Keine Berechtigung`  
`make: *** Keine Ziele. Schluss.`  
Zugriffsrechte überprüfen/setzen
- ▷ Erfolg: Es werden die möglichen Make-Optionen angezeigt. Auszug:

```

# -----
#
# Make Aufrufe für Projekt /c auf PC und Pi
# 2023-04-05
# -----
# Im Projektordner c/
#
# make           | Diese Anleitung anzeigen
# make chelp     | ▶ Startpunkt: Programmhilfe 'chelp' und
#               | alle Bibliotheken neu compilieren.
#               | 'chelp' starten und Einstellungen prüfen
#
# make infosys  | ▶ Systemeinstellungen anzeigen und verwalten
# make newprg   | ▶ Programm zum Anlegen neuer Projekte
#
# make picamctl | ▶ Pi: Camera-Steuerung picamctl
# make allpi    | Pi: Programme mit GPIO aus c/pi/
#
# make clean    | Alle Programme und Bibliotheken löschen
# make libs     | Nur Bibliotheken erzeugen
# make allbin   | Bibliotheken und Programme für PC
#
# make all      | Alles: Libs, Bin, Tests, Demo
# ...
# -----

```

- ▷ Befehl `make clean` löscht zur Sicherheit alle Programmreste des Projekts.
- ▷ Befehl `make chelp`
  - Compiliert zuerst alle Bibliotheken neu und dann das Programm `chelp`.
  - Danach wird das Programm gestartet und führt einen ersten Startcheck durch.
  - Im Startcheck können fehlenden Einstellungen vorgenommen werden.

**Fehlerfall**

Wenn sich `chelp` nicht kompiliert lässt, sollte man die Zugriffsrechte prüfen und setzen:

```
cd /home/user/test/c
```

```
id
```

```
ls -l
```

```
sudo chown -vR user:user *
```

in den Projektordner `c/` wechseln

liefert User-Name und User-Gruppe: Beispiel `user:user`

zeigt Besitzer und Gruppe der Projektdateien

Option: Besitzer und Gruppe neu einstellen. Achtung: nur im Projektordner aufrufen!

## B| Projekthilfe chelp einrichten

Die Werkstatt [chelp](#) unterstützt die Entwicklung von C Programmen. Sie stellt vielfältige Informationen zum Projekt /c/, zu den Bibliotheken, zum Linux und zu den eigenen Programmen zur Verfügung.

Die Programme können einzeln kompiliert und im Ordner `~/bin` können Startlinks erstellt werden.

Unter dem Menüpunkt [Einstellungen](#) können die Projekteinstellungen geprüft oder festgelegt werden.

Anleitung siehe Doku '[clar\\_chelp.pdf](#)'

### chelp kompilieren

Ins Projektverzeichnis `/home/user/c` wechseln.

Mit Befehl `make chelp` werden zuerst die Bibliotheken und dann das Programm kompiliert und gestartet.

### Startcheck

Beim Start von [chelp](#) erfolgt automatisch ein Startcheck zu Überprüfung der Einstellungen.

Im Dialog [Einstellungen](#) kann dieser Startcheck manuell aufgerufen werden. Dort können die verschiedenen Einstellungen auch angepasst werden.

▷ **Projektumgebung prüfen:** Es wird der aktuelle Projektordner DIRC mit dem im Setup gespeicherten Ordner verglichen.

▷ **Pfad zum Projektordner c/ prüfen:** Schreibzugriff auf den Projektordner und Lesezugriff auf `chelp.c` prüfen.

▷ **Startlinks:** Der Linkordner `~/bin` wird angelegt/geprüft und der Startlink für `chelp` wird angelegt/geprüft. Variable `$PATH` wird beim normalerweise beim Anmelden automatisch ergänzt. Mit dem angelegten Startlink kann das Programm nun in jedem Ordner mit `chelp` werden.

▷ **Texteditor für Console und X-Terminal:** Die Verfügbarkeit des eingestellten Editors prüfen.

▷ **Texteditor für X:** Die Verfügbarkeit des eingestellten Editors prüfen..

▷ **Starter für X-Programme:** Die Verfügbarkeit des eingestellten Starters prüfen.

▷ **Entwicklungsumgebung für C Programme:** Die Verfügbarkeit der eingestellten IDE prüfen.

▷ **Rechte, User und Gruppen prüfen:** Einstellen unter [Einstellungen](#) > [Rechte, User ...](#)

▷ **Starter für Geany prüfen:** Unter [Einstellungen](#) > [Geany Startdateien](#) können die Startdateien für Geany der aktuellen Umgebung angepasste werden.

Der Startcheck sollte fehlerfrei ablaufen!

Fehler unter [Einstellungen](#) oder in der [Konfiguration](#) beheben und den Befehl [Startcheck](#) nochmals aufrufen.

Eine genaue Anleitung findet man in der Doku [chelp](#):

[c/1\\_Dokus/clar\\_chelp.pdf](#)

oder [www.projekt.at/programs/c/clar\\_chelp.pdf](http://www.projekt.at/programs/c/clar_chelp.pdf)

## Einstellungen

Einstellungen für Projekt c/ und die Projektumgebung prüfen/anpassen.

Befehl: **Einstellungen**

Die genaue Anleitung findet man in Doku chelp:

[c/1\\_Dokus/clar\\_chelp.pdf](#)

oder

[www.projektc.at/programs/c/clar\\_chelp.pdf](http://www.projektc.at/programs/c/clar_chelp.pdf)

```

chelp
-----
Einstellungen für Projekt c/

Einstellungen für Projekt c/ und die Projektumgebung prüfen/anpassen.
Der Startcheck sollte fehlerfrei ablaufen!

Startcheck für chelp und Projekt /c | Links und Hilfsprogramme für /c prüfen
Rechte, User und Gruppe einstellen | Für alle Projektdateien aktualisieren
Geany Startdateien anpassen       | Dateipfade zum Projektordner einstellen
Links in ~/bin prüfen             | Programmstartlinks prüfen/anlegen
Makefile für Programmordner anlegen | Makefile für alle Programme anlegen
Projektlinks prüfen               | Alle symbolischen Links unter c/ prüfen

Optionale Projekteinstellungen     | Libraries und Hilfsprogramme prüfen
Infos zu Programm chelp            | Aktuelle Programmvariablen anzeigen
Versionskontrolle                  | Programmversionen anzeigen

Konfiguration von chelp bearbeiten | Datei chelp.conf
Stichwortliste für chelp bearbeiten | Datei chelp.keys

Befehl | ESC ?

```

## Hilfe zum Projekt c/

Infos und Dokumentationen können über die Befehle **Readme's** und **Stichworte** abgerufen werden.

Befehl: **Readme's | Dokus | man Seiten**

```

chelp
-----
Projekt c/ | Readme's und Dokus

| Stichworte | Farben | www Links |

Readme's
Kurzanleitung für Projekt /c
Readme's aus Projekt c/ anzeigen
String in Readme's suchen

Dokus und Anleitungen
0 Übersicht | 1_read.me
1 Start     | clar_start
2 Chelp     | clar_chelp
3 Neues Programm | clar_projekt
4 Basis-Objekte | clar_objekte1
5 In/Output Objekte | clar_objekte2
6 Datendateien | clar_objekte3
Dokus aus c/1_Dokus

Linux
Linux man Seiten und Header
Linux Header anzeigen

Quit

```

Befehl: **Stichworte**

Über selbst definierte Stichwortlisten können Informationen abgerufen und Programme gestartet werden.

```

chelp
-----
Stichwortlisten
Befehle: Cursortasten, RETURN Details, ESC oder q

PDF-Dokus zu den Programmen
Programme in c/bin/
Alle Help-Dateien 1_read.me

Libraries: Alle Dokus/Header und Testprogramme
.....
Lib iocon.h : Ausgabefunktionen, printf, printBlock, Farben
Lib iocon.h : Eingabefunktionen, getTaste, read-Funktionen
Lib err.h   : Err-Objekt, Fehlerbehandlung
Lib utils.h : Temporäre Strings, Hilfsfunktionen
Lib files.h : Funktionen für Ordner und Dateien
Lib vars.h  : Var-Objekt, globale Laufzeit-Variablen am Heap
Lib array.h : Array-Objekt, Container für Strings und sonstige Objekte
Lib data.h  : Data-Objekt zum Speichern von Arrays
Lib script.h : Konfigurations- und Scriptdateien
Lib token.h : Tokens-Objekt, C-Scripte zerlegen
Lib boxmenu.h : Menüdefinitionen und Menudialoge
Lib boxdirwahl.h: Ordner-, Device- oder Dateiwahl-dialoge
Lib boxlst.h : Objekt-Listwahl und List-Dialog
Lib box.h   : Box-Objekt für alle Bildschirmboxen

```

Die genaue Anleitung findet man in Doku chelp:

[c/1\\_Dokus/clar\\_chelp.pdf](#)

oder [www.projektc.at/programs/c/clar\\_chelp.pdf](http://www.projektc.at/programs/c/clar_chelp.pdf)

## C) Programm infosys

Das Programm **infosys** liefert umfangreiche Systeminformationen zum Rechner, zu Linux und zur Systemwartung. Das Programm ist immer 'Work In Progress'.

### Programm 'infosys' mit **chelp** compilieren

Programm **infosys** mit **chelp** compilieren und einrichten.

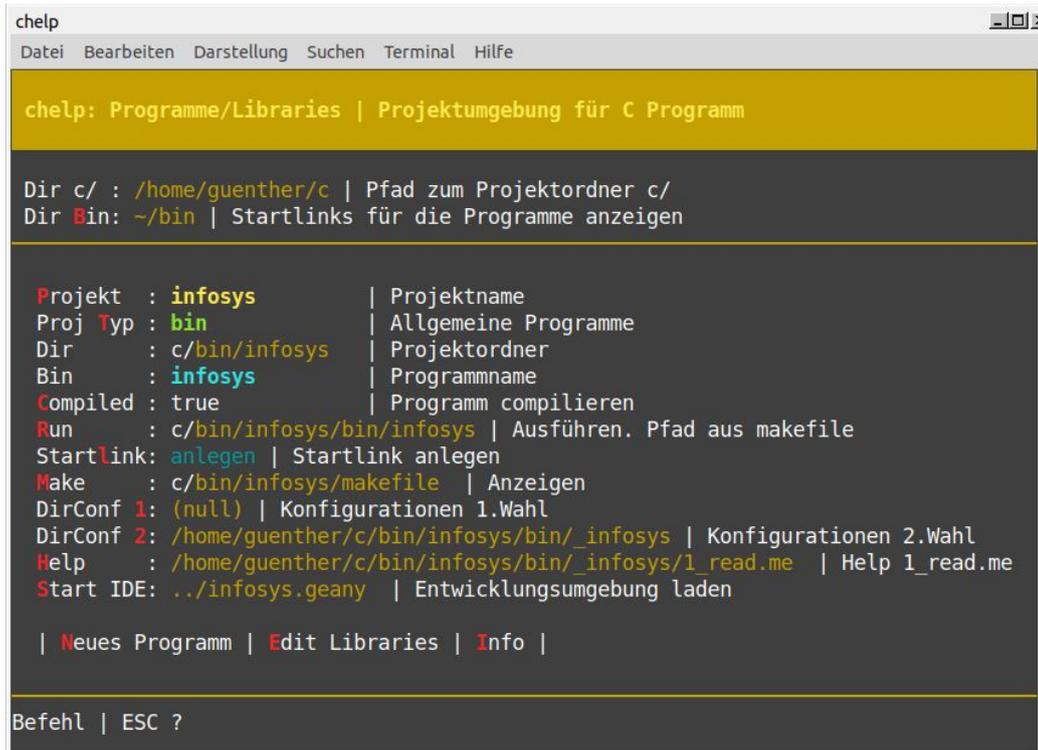
Im Menu von **chelp**: **Programme/Libraries anlegen/compilieren/ausführen**

Projektname **infosys** wählen: **Projekt | 1 bin | infosys**

Programm compilieren: **Compiled**

Startlink anlegen: **Startlink**

Programm kann dann einfach mit **infosys** gestartet werden.



```

chelp
Datei Bearbeiten Darstellung Suchen Terminal Hilfe

chelp: Programme/Libraries | Projektumgebung für C Programm

Dir c/ : /home/guenther/c | Pfad zum Projektordner c/
Dir Bin: ~/bin | Startlinks für die Programme anzeigen

Projekt : infosys           | Projektname
Proj Typ : bin              | Allgemeine Programme
Dir      : c/bin/infosys    | Projektordner
Bin      : infosys         | Programmname
Compiled : true             | Programm compilieren
Run      : c/bin/infosys/bin/infosys | Ausführen. Pfad aus makefile
Startlink: anlegen         | Startlink anlegen
Make     : c/bin/infosys/makefile   | Anzeigen
DirConf 1: (null)         | Konfigurationen 1.Wahl
DirConf 2: /home/guenther/c/bin/infosys/bin/_infosys | Konfigurationen 2.Wahl
Help     : /home/guenther/c/bin/infosys/bin/_infosys/1_read.me | Help 1_read.me
Start IDE: ../infosys.geany | Entwicklungsumgebung laden

| Neues Programm | Edit Libraries | Info |

Befehl | ESC ?

```

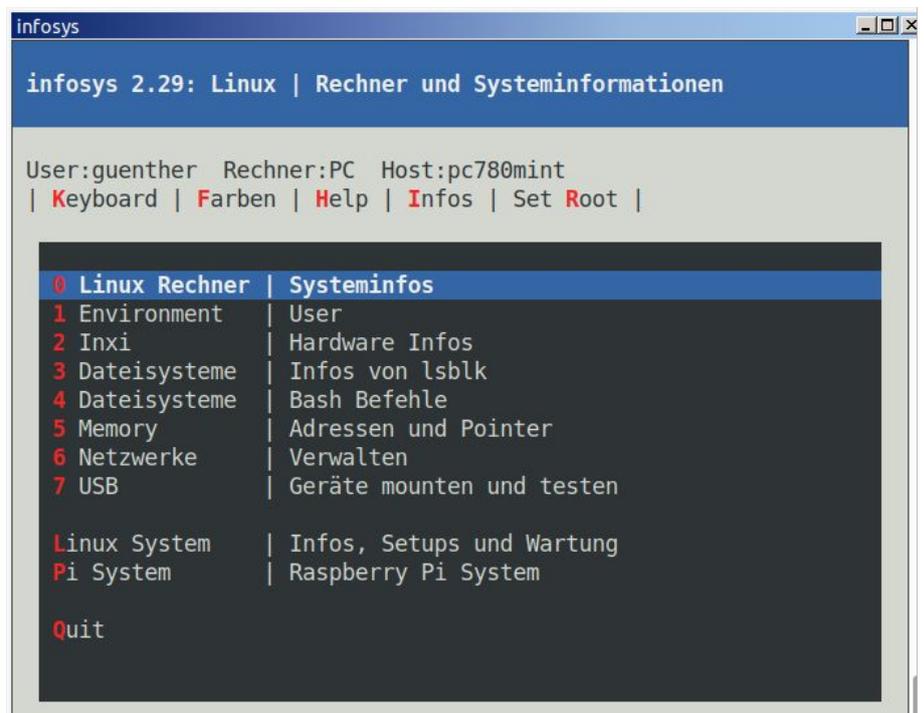
Entwicklungsumgebung: **Start IDE** Entwicklungsumgebung starten

Neues Programm: **Neues Programm** Programmgerüst für neues Programm anlegen

Anleitung für **chelp** siehe Doku chelp: [www.projektc.at/programs/c/clar\\_chelp.pdf](http://www.projektc.at/programs/c/clar_chelp.pdf)

### Programm infosys

Liefert ausführliche Informationen zum Rechner, zu Linux, zur Systemwartung und Systeminfos für den C Programmierer.



```

infosys

infosys 2.29: Linux | Rechner und Systeminformationen

User:guenther Rechner:PC Host:pc780mint
| Keyboard | Farben | Help | Infos | Set Root |

0 Linux Rechner | Systeminfos
1 Environment   | User
2 Inxi          | Hardware Infos
3 Dateisysteme  | Infos von lsblk
4 Dateisysteme  | Bash Befehle
5 Memory        | Adressen und Pointer
6 Netzwerke     | Verwalten
7 USB           | Geräte mounten und testen

Linux System    | Infos, Setups und Wartung
Pi System       | Raspberry Pi System

Quit

```

## Infos zum Keyboard

Programm `infosys` kann auch zum Überprüfen der Tastatureingaben verwendet werden.

Funktionierende Cursortasten sind für die Eingabefunktionen unbedingt notwendig.

Programm: `infosys` | `Keyboard`

```

guenther@pc780mint: ~
Datei Bearbeiten Darstellung Suchen Terminal Hilfe

Keyboard Definitionen für Projekt c/

Ein Tastendruck liefert ein oder mehrere Kbd-Bytes.
Diesen Bytefolgen wird im Programme ein uint16_t Code zugeordnet.

Kbd-Bytes mit maximal zwei Bytes liefern UTF8 Code. Für längere
Bytefolgen wird der Code aus Tabelle Kbd3Codes[] ermittelt.

UTF8 Codetabellen findet man in chelp.

Kbd-Bytes, Tastencode und Printcode anzeigen
Tabelle Printcodes
Tabelle Steuercodes
Test Steuercodes

Kbd-Bytes mit showkey anzeigen

Quit
  
```

Befehl: `infosys` | `Kbd-Bytes, Tastencode und Printcode anzeigen`

Alle Cursortasten prüfen!

Die von den Tasten gesendeten `Kbd-Bytes` können je nach System oder Terminal ( Console oder X-Terminal ) variieren.

In der Bibliothek `iocon.a` können verschiedene Bytefolgen für die Tasten definiert werden.

Siehe `c/lib/include/iocon.h`

```

guenther@pc780mint: ~
Datei Bearbeiten Darstellung Suchen Terminal Hilfe

Read Taste, Kbdcode, Tastencode und Printcode anzeigen

Kbd-Bytes | Tastencode | Printcode | Anzeige
-----|-----|-----|-----
Kbd 0x1b5b41 | Taste 0x0241 | Print no | 'taste_up'
Kbd 0x1b5b42 | Taste 0x0242 | Print no | 'taste_down'
Kbd 0x1b5b44 | Taste 0x0244 | Print no | 'taste_left'
Kbd 0x1b5b43 | Taste 0x0243 | Print no | 'taste_right'
Kbd 0x1b5b327e | Taste 0x02a2 | Print no | 'taste_ins'
Kbd 0x1b5b337e | Taste 0x02a3 | Print no | 'taste_entf'
Kbd 0x1b5b48 | Taste 0x0248 | Print no | 'taste_pos1'
Kbd 0x1b5b46 | Taste 0x0246 | Print no | 'taste_end'
Kbd 0x1b5b367e | Taste 0x02a1 | Print no | 'taste_pagedown'
Kbd 0x1b5b357e | Taste 0x02a0 | Print no | 'taste_pageup'
Kbd 0x40 | Taste 0x0040 | Print 0x40 | Zeichen '@'
Taste oder ESC ?
  
```

## Infos zum Rechner

Auszug der Systeminfos.

Wichtige Infos für Programmierer.

Diese Infos werden immer den aktuellen Erfordernissen entsprechend erweitert.

```

infosys
Static hostname: pc780mint
Icon name: computer-desktop
Chassis: desktop
Machine ID: 7a0781c5129d4f848f901e53f906eaf1
Boot ID: e20569177f644cda91467aedde57683a
Operating System: Linux Mint 20.2
Kernel: Linux 5.4.0-196-generic
Architecture: x86-64

Compiler | BitBreite | Byte Order im Speicher | sizeof()
-----|-----|-----|-----
Compiler: gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0
gnuc : 9.4.0 | Compiler Macros
Libc : 2.31 | Compiler Macros

getBitBreite() -> 64
isBigEndian() -> false | Little Endian LSB,MSB

sizeof(int32_t) -> 4 INT32_MAX : 2147483647
sizeof(uint32_t) -> 4 UINT32_MAX: 4294967295
sizeof(uint64_t) -> 8 UINT64_MAX: 18446744073709551615
sizeof(long) -> 8
sizeof(long unsigned int) -> 8
sizeof(long long unsigned int) -> 8
sizeof(void *) -> 8

Datum/Zeit | siehe lib 'datetime.h'
Maximales Datum -> 2106-02-07 07:28:15
Minimales Datum -> 1970-01-02 01:00:00
sizeof(time_t) -> 8 Bytes auf diesem System
Maximale DateTimeSec, time_t -> 4294967295 sec
Minimale DateTimeSec, time_t -> 86400 sec
Timeserver -> true
timedatectl status:
Local time: Di 2024-10-08 23:33:25 CEST
Universal time: Di 2024-10-08 21:33:25 UTC
RTC time: Di 2024-10-08 21:33:24
/tmp/less3956 lines 19-54/152 39%
  
```

## Infos zum Speicher

Memory-Map für die Speicher-Adressen von: C

```

infosys
Memory | Adressen und Pointer | freePtr()
Htmldatei erzeugen: aha -f tmp/lessXXX > info.html

Speicher-Adressen für C:
Stack      getMax() 0x7ffff9b54000
...
...      getStackMin() 0x7ffff9b33000
...
Heap      break 0x60b28e0bc000
...      getHeapMax() 0x60b28e0bc000
...      getHeapMin() 0x60b28e09b000
...      &end 0x60b28de76bf0
...
... uninitialized data
... data segment
... initialized data
...      &edata 0x60b28de75f38
...      &etext 0x60b28de5d621
...
... program text

Adressen für Pointer:
Test: Pointer p im Textsegment, am Stack und am Heap anlegen.
Funktion freePtr(p) ersetzt free() und kann mit beliebigen
Pointern verwendet werden.

Funktions Pointer:
runRunPtrTest =0x60b28de3de12 | isHeapPtr=0 isStackPtr=0

Fixe Pointer im data segment:
fix =0x60b28de649d4 "String fix im Textsegment!" | isHeapPtr=0 isStackPtr=0
sfix =0x60b28de649ef "String fix in Funktion!" | isHeapPtr=0 isStackPtr=0

Stack Pointer:
stack0 =0x7ffff9b51fb2 | Var aus main | isHeapPtr=0 isStackPtr=1
stack1 =0x7ffff9b51e30 "am stack1" | isHeapPtr=0 isStackPtr=1
stack2 =0x7ffff9b51e50 "am stack2" | isHeapPtr=0 isStackPtr=1

Heap Pointer:
heap =0x60b28e09d610 "String heap !" | isHeapPtr=1 isStackPtr=0
heap1=0x60b28e09d590 "String heap1!" | isHeapPtr=1 isStackPtr=0

Freigaben mit freePtr() und Rückgaben:
freePtr(fix ) -->fix
freePtr(stack1) -->stack1
freePtr(stack2) -->stack2
freePtr(sfix ) -->sfix
freePtr(NULL ) -->NULL
freePtr(heap ) -->NULL, free(heap) aufrufen
freePtr(heap1) -->NULL, free(heap1) aufrufen

Memory maps: cat /proc/5188/maps
    
```

Pointer anlegen und freigeben mit `freePtr()`:

Netzwerke:

Diese Befehle werden immer den aktuellen Erfordernissen entsprechend erweitert.

```

infosys

Netzwerke

Netz Device: enp2s0 | 00:23:54:c2:71:53 | 802-3-ethernet | Set Root

Ip | Netzinfos | Hostnamen | Geräte
device | Ein Netz Device wählen
Wlan Infos

NetworkManager | Netzwerk einstellen
wi ohne NetworkManager | Netzwerk einstellen

Hardware Infos

Ping
Netinfos | Alter Befehl ifconfig
Netzwerk neu starten

Quit
    
```

```

infosys

/dev/sdg disk BUS:usb SIZE:0B MODEL:USB MS Reader
/dev/sdh disk BUS:usb SIZE:931,5G MODEL:WDC WD10SMZW-
/dev/sdh1 part removable BUS:usb FSTYPE:vfat SIZE:33M
LABEL:BOOT
UUID :526C-868A
MOUNT:/boot/efi
/dev/sdh2 part removable BUS:usb FSTYPE:ext4 SIZE:195,5G
LABEL:Linux
UUID :b2206c98-06f9-4bb0-a4bc-362113d6e6b4
MOUNT:/
/dev/sdh3 part removable BUS:usb FSTYPE:ext4 SIZE:736,1G
LABEL:Daten
UUID :db68665e-d203-4999-9887-babf515c3fae
MOUNT:/media/test/Daten

/dev/sr0 rom BUS:ata SIZE:1024M MODEL:HL-DT-STDVD-RAM GH22LS30

Ende mit q

/tmp/less5188 lines 38-58/58 (END)
    
```

Infos zu den Dateisysteme

## D) Ein eigenes C Programme erstellen

Zum Testen von Projekt c/ sollte man dann probeweise ein eigenes Programm mit `newprg` aus einer Vorlage erstellen. Das Erstellen eines neuen Programms/Projekts aus Vorlagen wird in [clar\\_projekt.pdf](#) genau beschrieben.

Eine Kurzfassung:

### Hilfs-Programm 'newprg' compilieren

`chelp` starten. Im Menu von `chelp`: **Programme/Libraries anlegen/compilieren/ausführen**

Projektname `infosys` wählen: **Projekt | 1 bin | Projektordner: newprg**

Programm compilieren: **Compiled**  
Startlink anlegen: **Startlink**

```
chelp
-----
chelp: Programme/Libraries | Projektumgebung für C Programm

Dir c/ : /home/guenther/c | Pfad zum Projektordner c/
Dir Bin: ~/bin | Startlinks für die Programme anzeigen

Projekt : newprg           | Projektname
Proj Typ: bin             | Allgemeine Programme
Dir      : c/bin/newprg   | Projektordner
Bin      : newprg         | Programmname
Compiled : true           | Programm compilieren
Run      : c/bin/newprg/bin/newprg | Ausführen. Pfad aus makefile
Startlink: anlegen       | Startlink anlegen
Make     : c/bin/newprg/makefile | Anzeigen
DirConf 1: (null)        | Konfigurationen 1.Wahl
DirConf 2: /home/guenther/c/bin/newprg/bin/_newprg | Konfigurationen 2.Wahl
Help    : /home/guenther/c/bin/newprg/bin/_newprg/1_read.me | Help 1_read.me
Start IDE: ../newprg.geany | Entwicklungsumgebung laden

| Neues Programm | Edit Libraries | Info |

Befehl | ESC ?
```

### Neues Programm 'newprg' erstellen

Mit Befehl **Neues Programm** kann `newprg` gestartet werden. Programm `chelp` wird dabei durch `newprg` ersetzt

```
guenther@pc780mint: ~/c
newprg[0.18] Neues C-Programm mit Projekt c/ anlegen

Ein neues C Programm aus einer Vorlage erzeugen.
Es werden die Bibliotheken aus Projekt c/ verwendet!

Die gewünschten Vorgaben werden nun abgefragt.
Mit ESC können alle Eingaben abgebrochen werden!

Hilfe | Info | Chelp | Quit | RETURN ?
```

Die Vorlage `myprog_con` erzeugt die Grundlage für ein Terminal-Programm

```
guenther@pc780mint: ~/c
Projektvorlage wählen

Vorlagen aus: /home/guenther/c/vorlagen
Standard      : myprog_con

Vorlage wählen | Hilfe zur Vorlage
myprog_bsp/
myprog_con/
myprog_libtest/
myprog_min/
```

Programmname festlegen

```
guenther@pc780mint: ~/c
Programm- und Projektname

Gültige Dateinamen ohne Blanks und Sonderzeichen!
Bei libtest-Programmen wird der Name mit 'test' erweitert.

Programmname: x
```

Der Modus: **automatisch** oder **Weiter mit Taste**

```
guenther@pc780mint: ~/c
C-Projekt aus Vorlage anlegen

0 Projekt c/ : /home/guenther/c
1 Vorlage   : /home/guenther/c/vorlagen/myprog_con
2 Projektname: x   Programmname: x
3 Zielordner : /home/guenther/c/bin/x

Weiter mit: Modus

Step: 3 | Aendern | ESC | RETURN | ?
```

Danach landet man wieder in `chelp`. Mit Befehl **Programme/Libraries anlegen/compilieren/ausführen** kann das Programm x dann compiliert und gestartet werden.

Für die weitere Bearbeitung sollte die Entwicklungsumgebung `geany` installiert sein. Eine genaue Beschreibung zum Erstellen neuer Programmen aus Vorlagen findet man in [clar\\_projekt.pdf](#).

## Programme compilieren

Für das Compilieren von Programmen gibt es mehrere Möglichkeiten.

### 1. Programme mit make compilieren

In jedem Programmordner befindet sich neben den Sourcedateien eine passende `makefile` Datei. Diese Datei wird mit den folgenden `make` Befehlen aufgerufen:

```
make clean // löscht alle Objektdateien
make // erzeugt das Programm im Ordner
```

Der Befehl `make` compiliert immer die Dateien im Ordner oder alle Programme in den darunter liegenden Ordnern. Beim compilieren mit `make` werden keine Startlinks in `/home/user/bin` angelegt. Die Programme müssen dann mit dem Programm-Pfad gestartet werden. Die Startlinks können mit `chelp` angelegt werden. Siehe weiter Unten.

Im Startverzeichnis `'c/'` liefert der Aufruf `make` eine Liste von einigen Möglichkeiten:

Achtung: Der Aufruf von `make` funktioniert nur, wenn alle externen Bibliotheken für ein Programm installiert sind. Diese Bibliotheken können in `chelp` unter `Einstellungen/Optionale Projekteinstellungen` nachgeschlagen werden. Beispiel: Wenn `libjpeg-dev` für `jpg` Bilder nicht installiert ist, bricht `make allbin` bei Programm `pshow` ab.

```
#
# -----
# Make Aufrufe für Projekt /c auf PC und Pi
# 2023-04-05
# -----
# Im Projektordner c/
#
# make           | Diese Anleitung anzeigen
# make chelp     ▶ | Startpunkt: Programmhilfe 'chelp' und
#               | alle Bibliotheken neu compilieren.
#               | 'chelp' starten und Einstellungen prüfen
#
# make infosys  ▶ | Systemeinstellungen anzeigen und verwalten
# make newprg   ▶ | Programm zum Anlegen neuer Projekte
#
# make picamctl ▶ | Pi: Camera-Steuerung picamctl
# make allpi    | Pi: Programme mit GPIO aus c/pi/
#
# make clean    | Alle Programme und Bibliotheken löschen
# make libs     | Nur Bibliotheken erzeugen
# make allbin   | Bibliotheken und Programme für PC
#
# make all      | Alles: Libs, Bin, Tests, Demo
#
# make saveit   | Dateien und Ordner auf PC mit Pi synchronisieren
# make pshow    | Bilder und Videos auf der Console anzeigen mit fbp
# make kbdctl   | Songverwaltung für Yamaha PSR-S975
# make mtrainer | ALSA-MIDI Test und Gehörtraining
# make mid2midraw | ALSA-MIDI Gehörtraining
#
# -----
# Einzelne Programme compilieren
# In allen Unterordnern von c/ gibt es makefiles:
#
# make          | Programm(e) erzeugen
# make clean    | Programm(e) löschen
#
```

### 2. Programme mit chelp compilieren

Programme können auch mit `chelp` compiliert werden. Dabei werden alle Projekteinstellungen angezeigt und es können auch die Startlinks erstellt werden.

Siehe Beispiel im nächsten Kapitel: [Programm 'infosys' mit chelp erstellen](#)

Die genaue Anleitung für `chelp` findet man in Doku [clar\\_chelp.pdf](#):

[c/1\\_Dokus/clar\\_chelp.pdf](#)  
oder [www.projektc.at/programs/c/clar\\_chelp.pdf](http://www.projektc.at/programs/c/clar_chelp.pdf)

### 3. Programme mit Entwicklungsumgebung geany compilieren

Das Entwickeln und Compilieren von Programmen wird am besten über eine Entwicklungsumgebung (z.B. `geany`) vorgenommen.

Die genaue Anleitung findet man in Doku [clar\\_projekt.pdf](#)

[c/1\\_Dokus/projekt .pdf](#)  
oder [www.projektc.at/programs/c/clar\\_projekt.pdf](http://www.projektc.at/programs/c/clar_projekt.pdf)

### Fehler beim Compilieren

Versuch 1:

1. Prüfen, ob das aktuelle Verzeichnis Projektordner `c/` ist.
2. Alles bereinigen: Befehl `make clean`.
3. Bibliotheken `make libs` zuerst compilieren.
4. Befehl `make` wiederholen.

Versuch 2: Programme mit `chelp` einzeln compilieren

Versuch 3: Mit `help` die Entwicklungsumgebung `geany` mit dem Programmcode aufrufen und compilieren. Dabei werden die Fehler in den Source-Texten angezeigt.

## Datei- und Ordnerübersicht

```

Projekt c/: Datei- und Ordnerübersicht von c/.
Alle Programmordner haben denselben Aufbau wie 'help'.

```

c/	Projektordner. Verwendet relative symbolische Links
1_read.me	Diese Datei. Einstiegshilfe (obligatorisch!)
2_todo.txt	Hinweise zu Projekten und nicht behobene Fehler
makefile	Einstellungen für Befehl 'make' (obligatorisch!) make           ▶ Liefert Anleitung für mögliche make Aufrufe make clean   ▶ Löscht alle Programme von Projekt c/ make allbin  ▶ Alle Bibliotheken und PC-Programme erzeugen make allpi  ▶ Alle Linkbibliotheken und Programme aus c/pi/ erzeugen
GNU_Licence	GNU Lizenz (obligatorisch!)
<hr/>	
1_Dokus/	<b>Dokumentationen zu Projekt c/ (obligatorisch!)</b> siehe 1_Dokus/1_read.me
clar_vorwort.odt	Vorwort für Projekt c/
clar_vorwort.pdf	
clar_start.odt	Anleitung für Projekt c/
clar_start.pdf	
clar_chelp.odt	Beschreibungen für chelp
clar_chelp.pdf	
...	Weitere Anleitungen
<hr/>	
arduino/	Arduino Programme und PC/Pi Programme mit serieller Verbindung zum Arduino
1_read.me	Beschreibungen für Arduino/
...	siehe 1_read.me
<hr/>	
bin/	<b>Allgemeine Programme für Pi/PC Sourcen und Binaries (obligatorisch!)</b>
1_read.me	Beschreibung der Programme im Ordner bin/
makefile	Compiliert alle Programme in bin/
<hr/>	
chelp/	Aufbau der Programmordner am Beispiel chelp (obligatorisch)
makefile	Befehl 'make' compiliert 'chelp'
chelp.h	Hauptheader : glob. Includes, Konstanten, Deklarationen
chelp.c	Hauptprogramm: Menu(), Init(), Main() und Exit()
run.c	Hauptfunktionen
...	Weitere Header und Sourcen für chelp
...	Hilfsmodule und Funktionen
bin/	Programm und Konfigurationen
_chelp/	Konfigurationsordner
1_read.me	Kurzbeschreibung von chelp
chelp.conf	Konfigurationsdatei
chelp.keys.bak	Sicherungsdatei
chelp.keys	Stichwortliste
chelp.keys.bak	Sicherung
...	
chelp	Binary - fertiges Programm
...	Alle weitere Programme haben denselben Aufbau wie von chelp/
...	...
<hr/>	
bindemo/	Fertige Demo Programme (optional) Sourcen und Binaries
1_read.me	Beschreibung der Programme im Ordner bindemo/
makefile	für alle Programme unter bindemo/
...	demo00
...	...
<hr/>	
bsp/	Einfache C Beispiele zum Testen von speziellen Fragestellungen. Sourcen und Binaries. (optional)
1_read.me	Dokumentationen für die C Beispiele/Testprogramme
makefile	für alle Programme unter bsp
...	weitere Beispiele siehe 1_read.me
...	

Fortsetzung auf der nächsten Seite.

Fortsetzung: Projekt c/: Datei- und Ordnerübersicht von c/.

lib/	<b>Statische Bibliotheken von Projekt c/ für PC/Pi und Arduino. Quellen und Binaries. (obligatorisch!)</b>
l_read.me	Beschreibungen und Dokus für die statischen c/ Libraries
makefile	für alle Bibliotheken unter lib
libcomapi.a	Lib: ser. Kommunikation zwischen PC/Pi und Arduino
libiocon.a	Lib: IO-Funktionen fürs Terminal
libutils.a	Lib: Hilfsfunktionen für Strings, Shellbefehle, usw.
libvars.a	Lib: Strukturen für globale Variablen
libcomapi.geany	Starter für IDE geany mit Library comapi
libiocon.geany	Starter für IDE geany mit Library iocon
libutils.geany	Starter für IDE geany mit Library utils
libvars.geany	Starter für IDE geany mit Library vars
<hr/>	
Allgemeine statische c/ Bibliotheken für PC/Pi	
include/	Header *.h für alle Bibliotheken
iocon/	Lib: IO-Funktionen fürs Terminal
utils/	Lib: Hilfsfunktionen für Strings, Shellbefehle, usw.
vars/	Lib: Strukturen für globale Variablen Konfigurationsdateien lesen oder schreiben
comapi/	Lib: ser. Kommunikation zwischen PC/Pi und Arduino
...	
libraries/	Bibliotheken für Arduino IDE
com/	Lib: Arduino/PC/Pi Header für comapi
comapi/	Lib: ser. Kommunikation zwischen PC/Pi und Arduino
taste/	Lib: Allgemeine Tastenabfrage
kty81/	Lib: Temperatursensor kty81
mpc9801/	Lib: I2C Temperatursensor
...	
<hr/>	
libtest/	Testprogramme und Dokumentation für die Funktionen in den Libraries. Quellen und Binaries (optional)
l_read.me	Dokumentationen für Testprogramme
makefile	für alle Programme unter libtest
test_*/	Ordner für Doku- oder Testprogramm *
test_*.geany	Starter für IDE geany mit Source *
test*	Compiliertes Programm ohne '_'
...	
<hr/>	
pi/	Programme für Pi mit GPIO Ansteuerungen Quellen und Binaries (optional)
l_Dokus/	Entwicklungs-Dokus für Pi-Programme
bin/	fertige Pi-Programme
picamctl/	Camera Steuerung für Raspberry Pi
piclock/	Echtzeituhr PCF8583 für Raspberry Pi
...	
picamctl.geany	Starter für IDE geany
piclock.geany	Starter für IDE geany
...	
bininc/	Linkbibliothek für Steuerungs-Module. Diese Module werden über symbolische Links eingebunden.
events	Verwaltung und Ansteuerung von GPIO-Geräten
gpio	GPIO-Module für die GPIO-Hardware vom Pi
bmp180	Druck und Temperatursensor bmp180. Module für Schnittstelle gpioi2.c und SBus i2c-Interface
...	
bsp/	Testbeispiele
l_read.me	Infos zu Pi-Programmen
makefile	für alle Pi-Programme
<hr/>	
vorlagen/	<b>Programm-Vorlagen für neue Projekte. Verwendung mit 'newprg' (obligatorisch!)</b>
myprog_con	Standardvorlage mit Menu
myprog_min	Minimalvorlage
myprog_libtest	Vorlage für Testprogramme
myprog_bsp	Vorlage für Testbeispiele
...	

## GNU General Public License

/\*

Copyright 2022-2025 Günther Schardinger <schardinger@projektc.at>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

---

Dieses Programm ist Freie Software: Sie können es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Wahl) jeder neueren veröffentlichten Version, weiter verteilen und/oder modifizieren.

Dieses Programm wird in der Hoffnung bereitgestellt, dass es nützlich sein wird, jedoch OHNE JEDE GEWÄHR,; sogar ohne die implizite Gewähr der MARKTFÄHIGKEIT oder EIGNUNG FÜR EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License für weitere Einzelheiten.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe <<https://www.gnu.org/licenses/>>.

\*/