

Start CLAR | C - Linux - Arduino - Raspberry

Inhaltsverzeichnis

Start CLAR | C - Linux - Arduino - Raspberry

Projekt c/ einrichten

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

Raspberry Pi

1. Download	3
2. Entpacken	3
3. chelp einrichten	3

Linux PC

1. Download	4
2. Einrichten	4
3. chelp einrichten	4

Projekthilfe chelp einrichten

chelp kompilieren	5
infosys kompilieren	5

Keyboard testen

Programm infosys	6
Testprogramm für Tastencodes	6

Programme compilieren

Makefiles	7
Fehler beim Compilieren	7

Eigene C Programme erstellen

GNU General Public License

Projekt c/ einrichten

Homepage und Downloads:

<http://www.schmuckhexen.at/programms>

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/ .

Die Lage des Projektordners c/ ist beliebig. Der Ordner c/ kann auf internen oder externen (z.B USB) Dateisystemen mit symbolischen Links 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 Beschreibungen

Projekt c/ einrichten. Die ersten Schritte:	www.schmuckhexen.at/programs/c/clar_start.pdf
Projekthilfe und Projektmanager:	www.schmuckhexen.at/programs/c/clar_chelp.pdf
Ein neues C Programm erstellen:	www.schmuckhexen.at/programs/c/clar_projekt.pdf
Basisobjekte ohne Terminal In/Ouput:	www.schmuckhexen.at/programs/c/clar_objekte1.pdf
Terminalsteuerung Box-Objekte für In/Ouput:	www.schmuckhexen.at/programs/c/clar_objekte2.pdf

Dokumentation: C-Programme und Bibliotheken

Die Dokumentationen für C-Programme/Bibliotheken befinden sich in Hilfedateien '1_read.me' und in den C-Headern der Programme/Bibliotheken.

Hilfe zu den fertigen Programmen liefert die Startoption '-h'.

Einstiege:

▷ Programm chelp	Menügesteuerter Zugriff auf alle Dokus
▷ Projekt c/ Einstieg und Übersicht	c/1_read.me
▷ Header/Dokus für Bibliotheksfunktionen	c/lib/1_read.me
▷ Testprogramme für Bibliotheksfunktionen	c/libtest/1_read.me

Raspberry Pi

User: pi
 Home: /home/pi
 Projektordner: /home/pi/c/ Das Standard Projektverzeichnis!
 Für andere Verzeichnisse siehe Anleitung Linux PC!

Homepage: <http://www.schmuckhexen.at/programs/c/index.html>
 Download Ziel: /home/pi/c/c.tar.gz
 StartLinks: /home/pi/bin

1. Download

Projektordner /home/pi/c/ anlegen. Danach wird das Projektarchiv [c.tar.gz](http://www.schmuckhexen.at/programs/c/c.tar.gz) von der Homepage <http://www.schmuckhexen.at/programs/c/index.html> in den Projektordner /home/pi/c/ geladen.

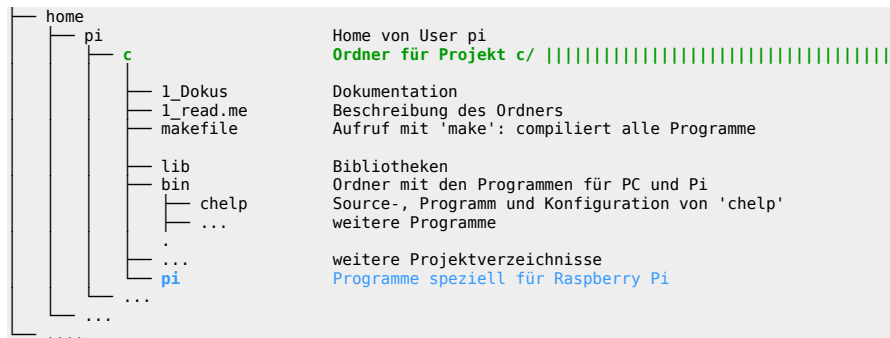
2. Entpacken

In der Console oder im X-Terminal:

Ins Projektverzeichnis wechseln: `cd /home/pi/c`
 Archiv entpacken : `tar -xzf c.tar.gz -C /home/pi`

Damit werden die Archivdateien ins Projektverzeichnis /home/pi/c/ wie folgt kopiert:

Die Dateien am Raspberry Pi:



3. chelp einrichten

Im Projektordner `home/pi/c` den Befehl `make chelp` aufrufen.

Das Programm wird compiliert und gestartet. Im Startcheck können die fehlenden Einstellungen vorgenommen werden.

Anleitung siehe Doku chelp: www.schmuckhexen.at/programs/c/clar_chelp.pdf

Linux PC

Lage des Projektordners ist beliebig!

Installationsbeispiel:

```
User:          maier
Home:         /home/maier
Projektordner: /media/maier/EXT4/c      Projektordner c/ auf USB-Festplatte
Download:     /media/maier/EXT4/c/c.tar.gz
Startlinks:   /home/maier/bin
```

1. Download

Das Projektarchiv [c.tar.gz](http://www.schmuckhexen.at/programs/c/index.html) wird von der Homepage <http://www.schmuckhexen.at/programs/c/index.html> in den Projektordner `/media/maier/EXT4/c` geladen.

2. Einrichten

Damit der Projektordner `c/` auf allen Systemen immer gleich ansprechbar ist, wird ein symbolischer Link `/home/pi` verwendet.

Symbolischen Link `pi` in `/home` auf das Elternverzeichnis `/media/maier/EXT4` von `c/` anlegen:

Das Anlegen des Links `pi` erfordert root-Rechte. `su` oder `sudo` verwenden!

In den Ordner `/home` wechseln:

Den Zugriff auf Projektordner `c` mit `ls` testen

```
ls /media/maier/EXT4
```

```
c ...
```

Die Befehlszeile auf `ln.....` ändern

```
ln -is /media/maier/EXT4 pi
```

Kontrolle:

```
ls -l pi
```

```
lrwxrwxrwx 1 root root 14 Feb 18 2015 pi -> media/maier/EXT4
```

Die weiteren Schritte erfolgen wie am Raspberry Pi:

Ins Projektverzeichnis wechseln: `cd /home/pi/c`

Archiv entpacken: `tar -xzf c.tar.gz -C /home/pi` Ins Projektverzeichnis entpacken.

Damit werden die Archivdateien ins Projektverzeichnis `/home/pi/c/` wie folgt kopiert:

Die Dateien am PC:

```

.
├── home
│   ├── maier
│   │   ├── tmp
│   │   ├── bin
│   │   ├── ...
│   │   └── ...
│   └── pi -> media/maier/EXT4
└── media/maier/EXT4
    ├── c
    │   └── Ordner für Projekt c/ |||
    ├── l_Dokus
    │   └── l_read.me
    │       ├── Dokumentation
    │       └── Beschreibung des Ordners
    ├── lib
    │   ├── bin
    │   └── chelp
    │       ├── Bibliotheken
    │       └── Ordner mit den Programmen für PC und Pi
    │           ├── Source-, Programm und Konfiguration von 'chelp'
    │           └── ...
    └── pi
        └── weitere Projektverzeichnisse
            └── Programme speziell für Raspberry Pi
  
```

3. chelp einrichten

Im Projektordner `home/pi/c` den Befehl `make chelp` aufrufen.

Das Programm wird kompiliert und gestartet. Im Startcheck können die fehlenden Einstellungen vorgenommen werden.

Anleitung siehe Doku `chelp`: www.schmuckhexen.at/programs/c/clar_chelp.pdf

Projekthilfe chelp einrichten

Mit Programm `chelp` können Informationen zum Projekt `/c` und den eigenen Programmen abgefragt und verwaltet werden.

Anleitung siehe Doku '[clar_chelp.pdf](#)'

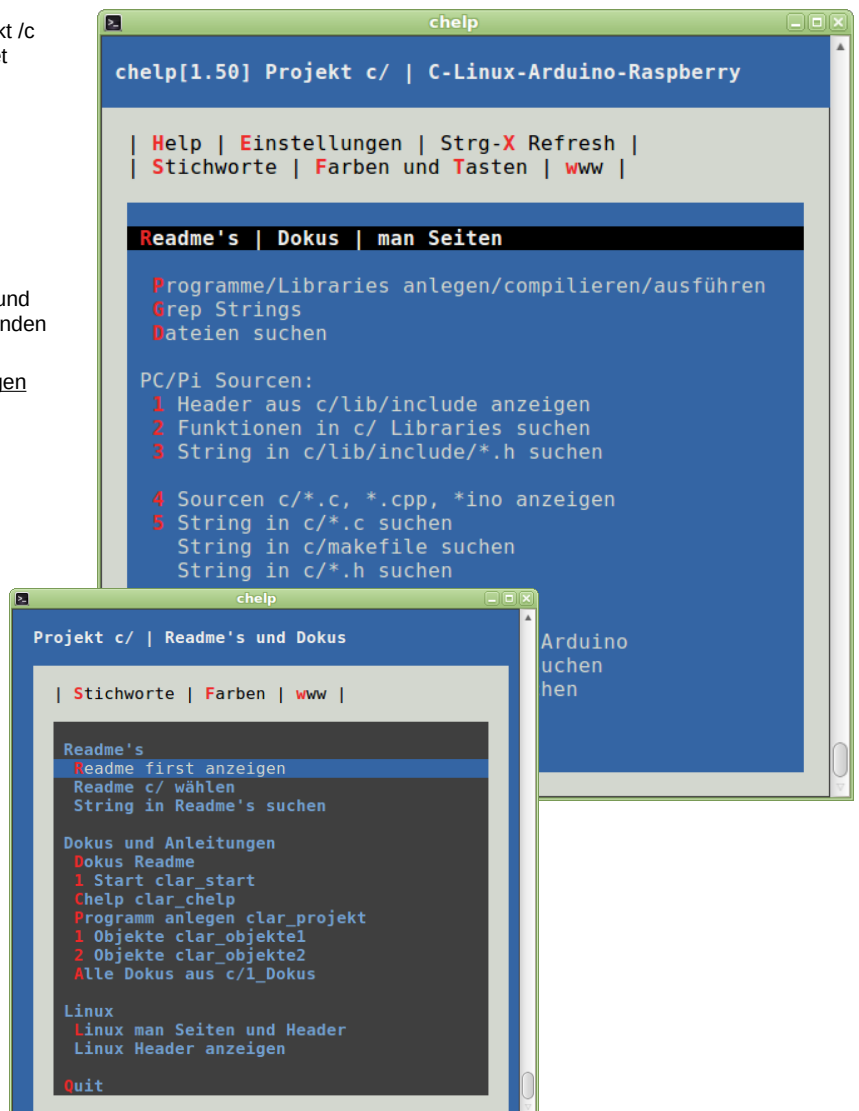
chelp kompilieren

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

Mit Befehl `make chelp` wird das Programm kompiliert und automatisch gestartet. Im Startcheck können die fehlenden Einstellungen vorgenommen werden.

Der Startcheck kann auch über den Dialog [Einstellungen](#) aufgerufen werden.

Das Programm `chelp` kann nun in jedem Ordner ohne Pfadangabe gestartet werden.



Beispiel: Dokumentationen anzeigen.

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

Anleitung siehe Doku `chelp`:

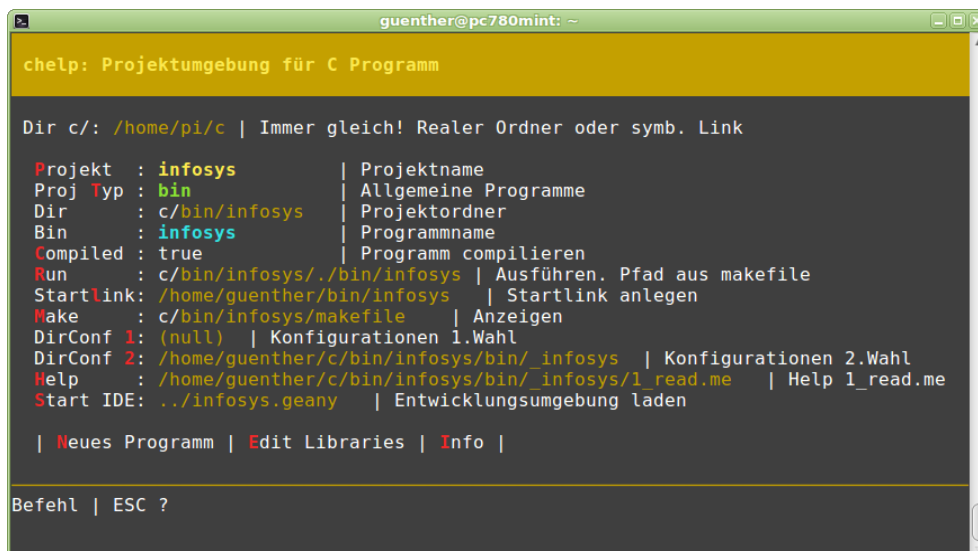
www.schmuckhexen.at/programs/c/clar_chelp.pdf

infosys compilieren

Beispiel: Programm `infosys` zum Anzeigen von Systeminformationen compilieren und einrichten.

Befehl: `Programme/Libraries anlegen/compilieren/ausführen`
 Projektname `infosys` wählen: `Projekt | 1 bin | infosys/`
 Programm compilieren: `Compiled`
 Startlink anlegen: `Startlink`

Das Programm kann dann in jedem Verzeichnis mit `infosys` gestartet werden.



Anleitung siehe Doku `chelp`: www.schmuckhexen.at/programs/c/clar_chelp.pdf

Keyboard testen

Zum Überprüfen der Tastatureingaben kann das Programm `infosys` verwendet werden.

Programm `infosys`

Programm `infosys` [compilieren](#).

```

guenther@pc780mint: ~
Datei Bearbeiten Darstellung Suchen Terminal Hilfe

infosys 1.81: Linux Systeminformationen

User:guenther Rechner:PC Host:pc780mint CPU:x86_64
| Keyboard | Farben | Help | Programinfos | Set root |

0 Linux Rechner | Systeminfos
1 Environment   | User
2 Inxi          | Hardware Infos
3 Dateisysteme  | Infos von lsblk
4 Dateisysteme  | Bash Befehle
5 Netzwerke     | Verwalten

Installation   | Grundinstallation
Wartung        | Systemwartung

9 Raspberry    | Wartung

```

Testprogramm für Tastencodes

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 ?

```

Programme compilieren

Programme können mit [chelp](#) oder dem Befehl [make](#) compiliert werden.

In jedem Fall wird ein Makefile 'makefile' verwendet. Für jedes Programm gibt es nur ein Makefile mit relativen Pfadangaben.

Makefiles

Beispiel: Programm 'infosys' im Ordner [c/bin/infosys/](#)

Das eigentliche Makefile ist [c/bin/infosys/makefile](#)

Makefile [c/bin/makefile](#) erzeugt alle Programme im Ordner [c/bin](#).

Makefile [c/makefile](#) erzeugt alle Programme des Projekts.

Beispiel: Makefile [c/makefile](#)

Ins Projektverzeichnis [/home/pi/c](#) wechseln und Befehl [make](#) aufrufen.

Es werden die Aufrufoptionen angezeigt:

```
# =====
# Make Aufrufe für Projekt /c Programme auf PC und Pi
# 2022-01-21
# -----
# Im Ordner c/
#
# make          // Diese Information anzeigen
# make clean    // Alles löschen
#
# make allbin   // Bibliotheken und Programme für Pi oder PC
# make libs     // Nur Bibliotheken erzeugen
#
# make chelp    // Hilfe für Projekt c/ compilieren
# make infosys // Programm für Systemeinstellungen
# make pshow   // Bilder und Videos auf der Console anzeigen mit fbp      #
# make saveit  // Dateien und Ordner auf PC mit Pi synchronisieren
#
# make kbdctl   // Songverwaltung für Yamaha PSR-S975
# make mtrainer // ALSA-MIDI Test und Gehörtraining
#
# make all      // Alles: Libs, Bin, Tests, Demo
# make Pi       // Pi Programme mit GPIO aus c/pi/
#
# -----
# Einzelne Programme compilieren
# In allen Unterordnern von c/ gibt es makefiles:
#
# make          // Program(e) erzeugen
# make clean    // Program(e) löschen
# =====
```

Beispiel: Befehl [make allbin](#) .z.B. erzeugt alle Bibliotheken und wichtige Programme.

```
Programme erstellen
make allbin
```

Die Makefiles kommentieren alle Aktionen mit Kommentarzeilen [# ...](#)

Zuerst werden die statischen Libraries [compiliert](#) und [erstellt](#).

Danach werden die Programme [compiliert](#) und [gelinkt](#).

```
make -C ./lib
make[1]: Entering directory `/home/guenther/c/lib'
#-----
make -C utils
make[2]: Entering directory `/home/guenther/c/lib/utils'
# ../include/
# compilieren ...
gcc -c -Wall -std=gnu99 -Di686 -I. -I../include/ -o "utils.o" "utils.c"
# ../include/ ...
...
# static lib erstellen ...
ar rcs ../libutils.a utils.o tmpstr.o files.o filesys.o err.o datetime.o
make[2]: Leaving directory `/home/guenther/c/lib/utils'
#-----
...
#-----
make -C ./vorlagen
make[1]: Entering directory `/home/guenther/c/vorlagen'
# compilieren ...
gcc -c -std=gnu99 -Wall -Di686 -I. -I../lib/include/ -I../lib/wiringpi_sim/ -o "myprog.o" "myprog.c"
# linken ...
gcc -o ./myprog myprog.o -L../lib/wiringpi_sim/ -lwiringPi -L../lib/ -liocon -L../lib/ -lutils -lm -L../lib/ -lvars
make[1]: Leaving directory `/home/guenther/c/vorlagen'
#-----
```

Befehl [make](#) bricht beim ersten Fehler ab!

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 [chelp](#) die Entwicklungsumgebung [geany](#) mit dem Programmcode aufrufen und compilieren.
Dabei werden die Fehler in den Source-Texten angezeigt.

Eigene C Programme erstellen

Das Erstellen eines neuen Programms/Projekts aus Vorlagen wird in [clar_projekt.pdf](#) beschrieben.

GNU General Public License

```
/*
 *
 * Copyright 2020 Günther Schardinger <v.schardinger@gmx.net>
 *
 * 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 2 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, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
 * MA 02110-1301, USA.
 */
```