Projekt c/ | C - Linux - Arduino - Raspberry

Inhaltsverzeichnis

Einleitung

Das Projekt c/ beschreibt Werkzeuge und Hilfsprogramme für komplexe Hard- und Softwareprojekte für PC, Arduino und Raspberry.
Es geht um Lösungen, die auch im Echtzeitbetrieb laufen und auch nach längerer Zeit noch gut gewartet und erweitert werden können.
Entwicklung und Wartung können auf jedem Linux-System oder über ssh und ssh -X durchgeführt werden.
Die Zielsysteme benötigen nur ein Linux Betriebssystem und eine USB Schnittstelle.

Homepage

Homepage und Downloads: www.schmuckhexen.at/programms

Allgemeine Beschreibungen

Die allgemeinen Dokumentationen findet man unter c/1_Dokus oder im Internet:

Vorwort:	www.schmuckhexen.at/programs/c/clar_vorwort.pdf	c/1_Dokus/clar_vorwort.pdf
Projekt c/ einrichten. Die ersten Schritte:	www.schmuckhexen.at/programs/c/clar_start.pdf	c/1_Dokus/clar_start.pdf
Projekthilfe und Projektmanager:	www.schmuckhexen.at/programs/c/clar_chelp.pdf	c/1_Dokus/clar_chelp.pdf
Ein neues C Programm erstellen:	www.schmuckhexen.at/programs/c/clar_projekt.pdf	c/1_Dokus/clar_projekt.pdf
Basisobjekte ohne Terminal In/Ouput:	www.schmuckhexen.at/programs/c/clar_objekte1.pdf	c/1_Dokus/clar_objekte1.pdf
Terminalsteuerung Box-Objekte für In/Ouput:	www.schmuckhexen.at/programs/c/clar_objekte2.pdf	c/1_Dokus/clar_objekte2.pdf

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 name/bin/_name/
- ▷ Die Dokumentation des Programmcodes befindet sich in den C-Headern der Programme oder Bibliotheken.
- ▷ Hilfe zu den fertigen Programmen liefert immer 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

IDE 'geany'

Entwicklungsumgebung für Pi und PC. Zu allen C-Programmen des Projekt c/ gibt es Startdateien *.geany.

Beispiel: Programm chelp, Startdatei c/bin/chelp.geany.
Wenn man die Entwicklungsumgebung für chelp über chelp.geany startet, dann werden die zuletzt verwendeten Dateien wieder geladen.

Bei einer Neuinstallation oder einer Verschiebung des Projektordners c/ stimmen die Pfade in chelp.geany nicht mehr!
Dieser Fehler kann in chelp mit dem Befehl Einstellungen > Geany Startdateien anpassen behoben werden.

Installation

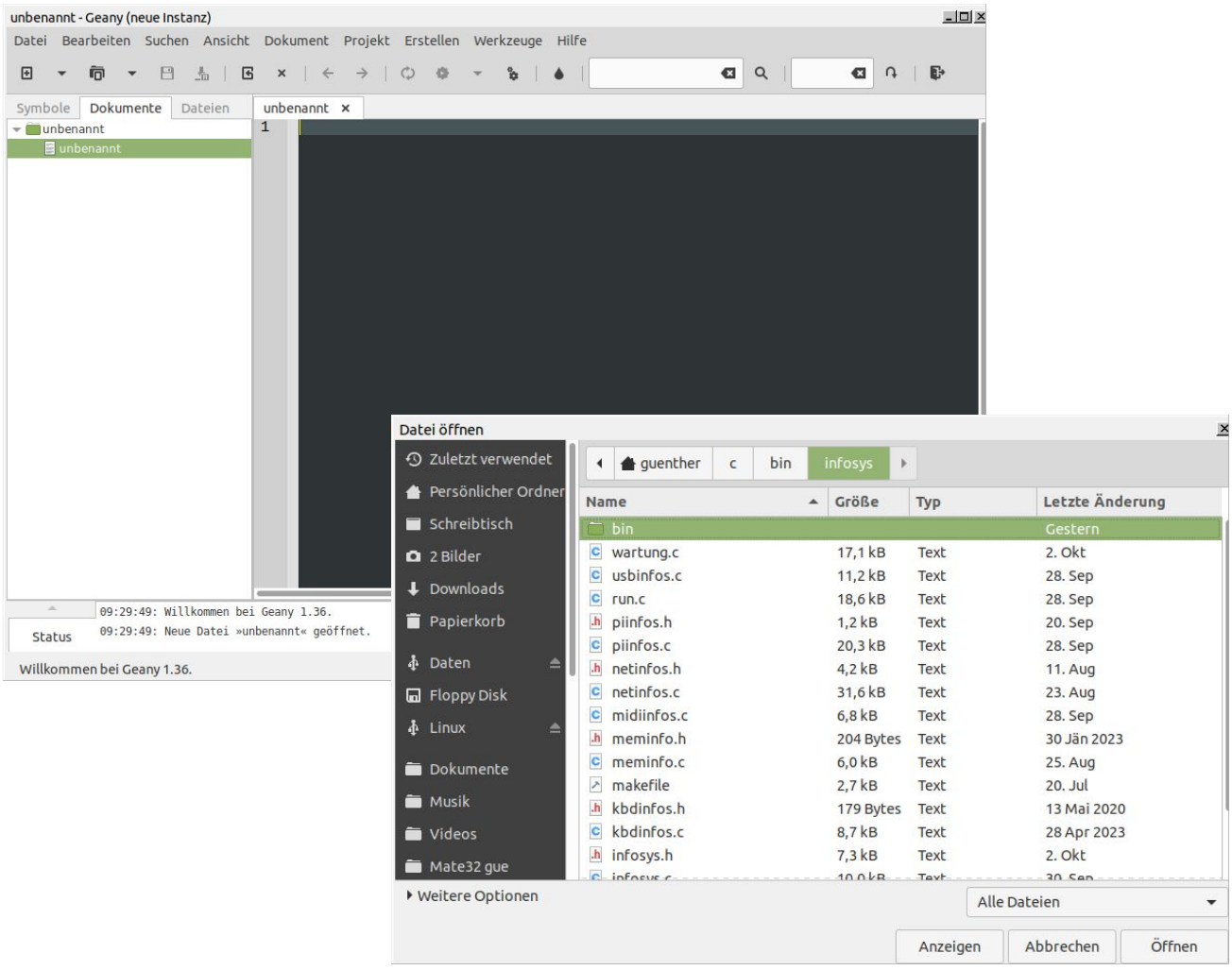
geany ist eine übersichtliche Entwicklungsumgebung für Pi und PC. Sie funktioniert auch sehr gut mit ssh -X am PC.
Die Bedienung ist einfach und der Funktionsumfang ist ausreichend. geany funktioniert auch mit großen Projekten.

Option: Installation von geany

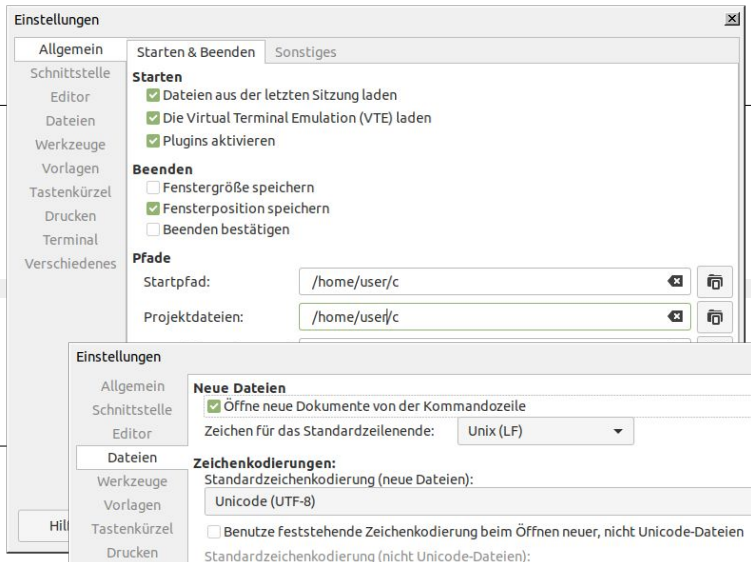
```
sudo apt-get install geany
```

Die Entwicklungsumgebung geany für das Programm xxx kann über chelp oder
den Projektstarter /home/pi/c/bin/xxx.geany gestartet werden.

Beim ersten Start müssen eventuell die Projektdateien makefile, xxx.h, xxx.c und xxx.c mit Datei/Öffnen angezeigt werden.



Konfiguration überprüfen



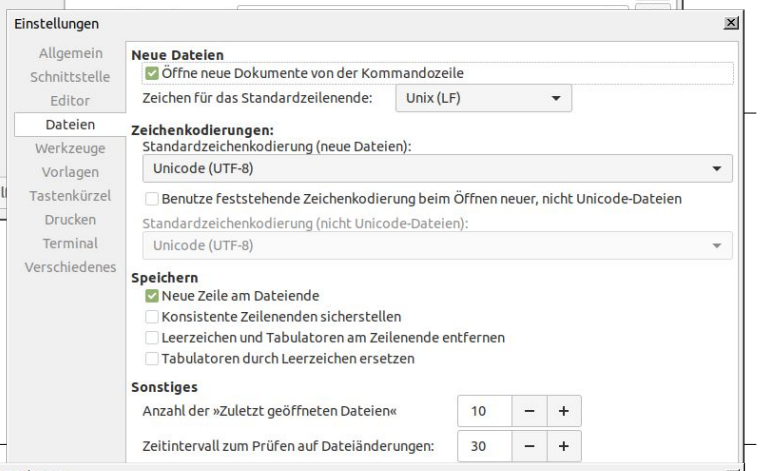
Einstellungen: Bearbeiten/Einstellungen

Reiter: Allgemein/Starten & Beenden

Pfade

Als Startpfad wird der Projektordner eingestellt:

```
/home/user/c/....
```



Bearbeiten/Einstellungen: Reiter: Dateien

Zeichenkodierungen

Standardzeichencodierungen: UTF-8
Benutze feststehende Zeichencodierung: ja

Speichern

Leerzeichen am Zeilenende entfernen

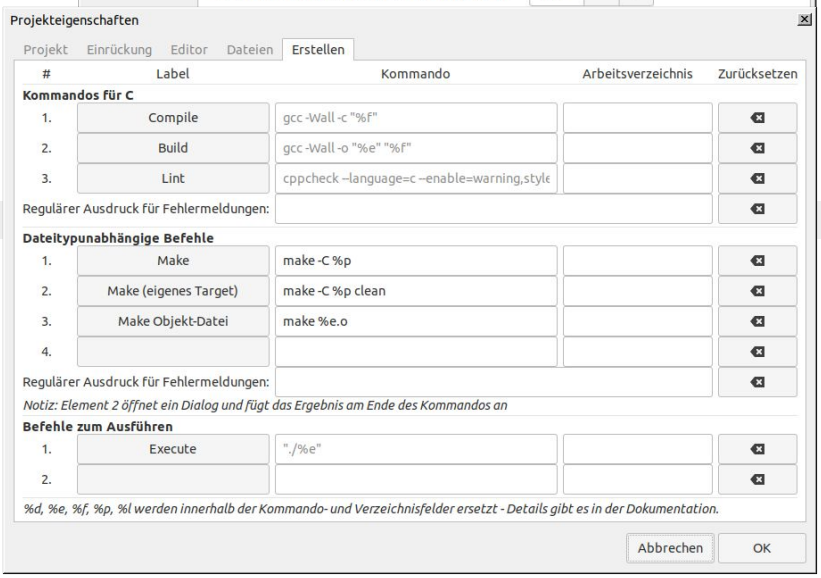
Make Einstellungen unter: Erstellen

Kommandos zum Erstellen konfigurieren

Dateitypunabhängige Befehle

Damit wird immer das makefile im passenden Verzeichnis verwendet.

```
make -C %p
make -C %p clean
```



Mit dieser Einstellung funktionieren die make Befehle im Menü oder mit den Tasten:

Erstellen/Make: oder Taste Umschalt-F9

Erstellen/Make (eigenes Target): oder Taste Umschalt-Strg-F9

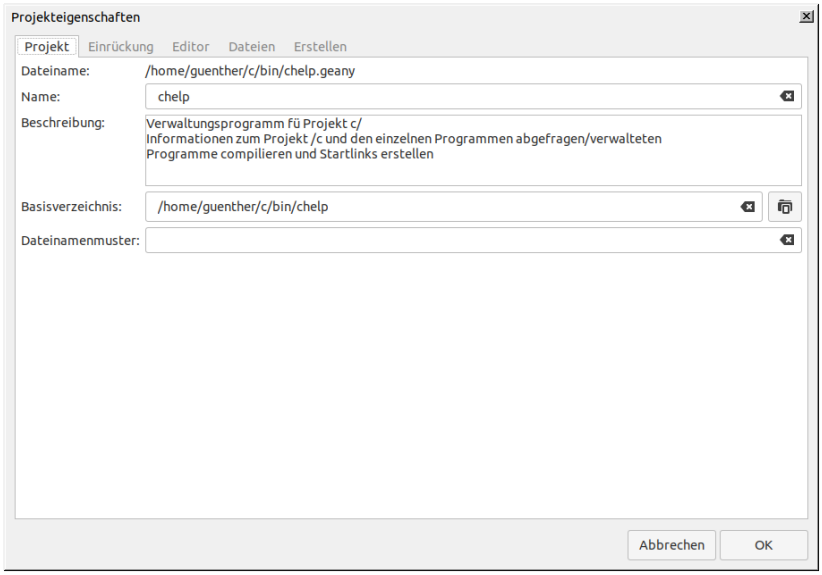
Erstellen/Ausführen: oder Taste F5
Programm ausführen

Danach Projekt/Eigenschaften aufrufen.

Beispiel: Programm chelp.

Die Projekteigenschaften werden in der Datei z.B. chelp.geany gespeichert.

Sie ermöglichen einen komfortablen Start der Entwicklungsumgebung mit allen verwendeten Programm-Soucen von chelp



Name: z.B chelp
Beschreibung: optional
Basisverzeichnis: /home/user/c/pi/bin/chelp

Achtung:

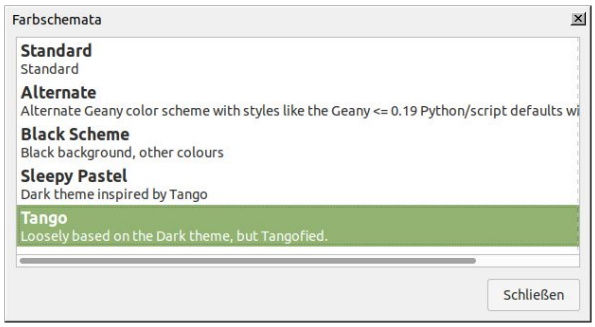
Bei einer Neuinstallation oder einer Verschiebung des Projektordners c/ stimmen die Pfade in chelp.geany dann nicht mehr!

Dieser Fehler kann in chelp mit dem Befehl Einstellungen > Geany Startdateien anpassen behoben werden.

Farbschemata

Geany Themes von
https://www.geany.org/download/themes/
herunterladen.

Das gewünschte Farbschema auswählen und nach
/home/guenther/.config/geany/colorschemes
kopieren und über Menüpunkt 'Ansicht' auswählen



Vorlagen

Für Befehl Datei/Neu (aus Vorlage)

Die eigenen Vorlagen können in Ordner /usr/share/geany/templates/files abgelegt werden:

```
├── file.html
├── file_html5.html
├── file.php
├── main.c          // Hauptprogramm
├── main.h          // Hauptheader
├── modul.h         // Header für Module
├── ...
└── NotUsed         // nicht benötigte Vorlagen
    ├── file.rb
    ├── ...
    .
    └── ...
```

Ein C Programm erstellen

Neue Programme können aus Vorlagen mit Programm newprg erzeugt werden.

Wenn newprg noch nicht compiliert/eingerichtet wurde, kann der Aufruf auch über chelp erfolgen.

newprg

Mit Programm newprg wird ein neues C Programm anlegt.

Standardvorlage myprog_con/ auswählen.

Mit Hilfe können Infos zur aktuellen Vorlage angezeigt werden.

Programmname wählen. Zum Beispiel: xxx

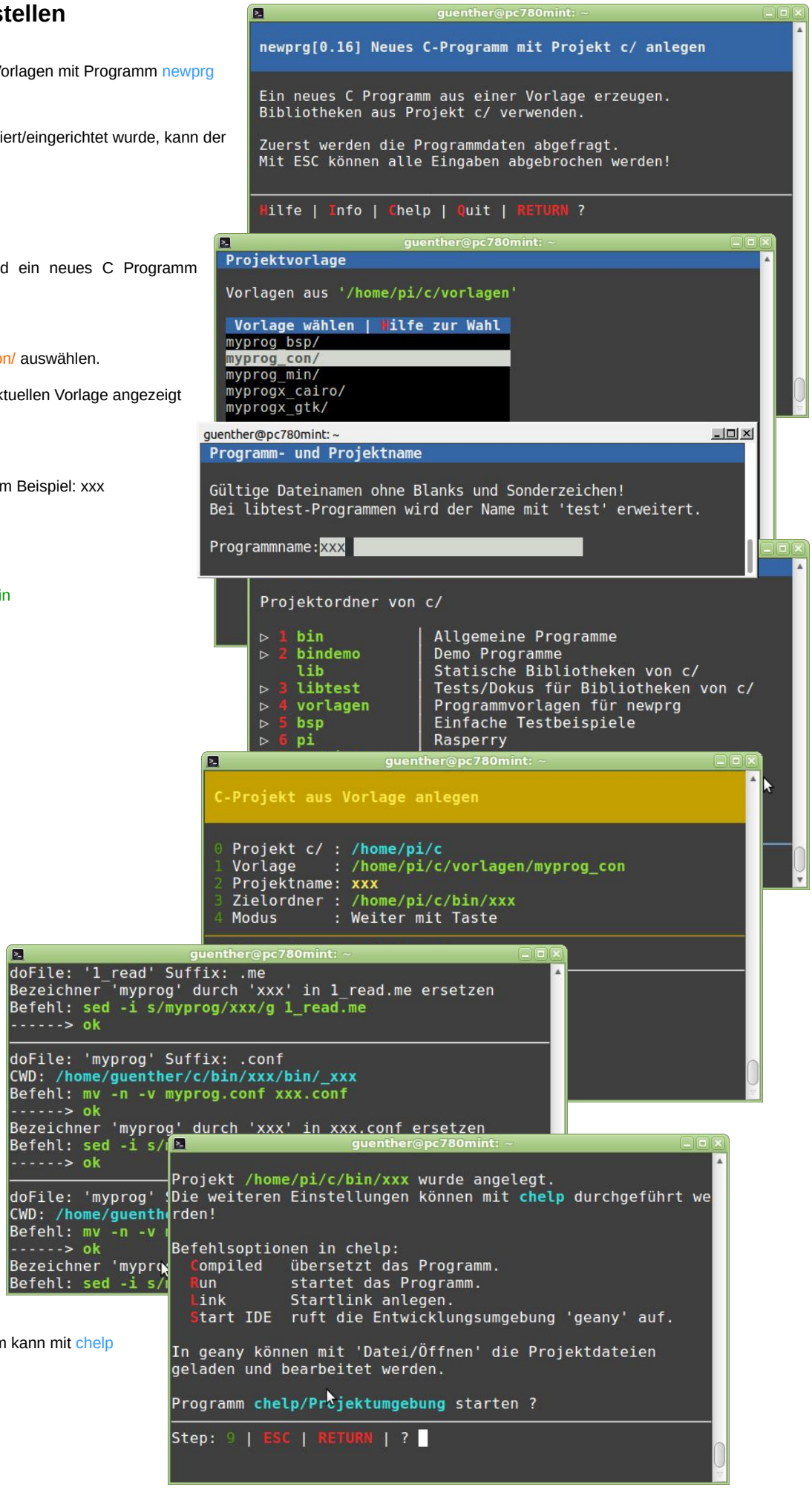
Standard Projektordner: 1 bin

Zusammenfassung

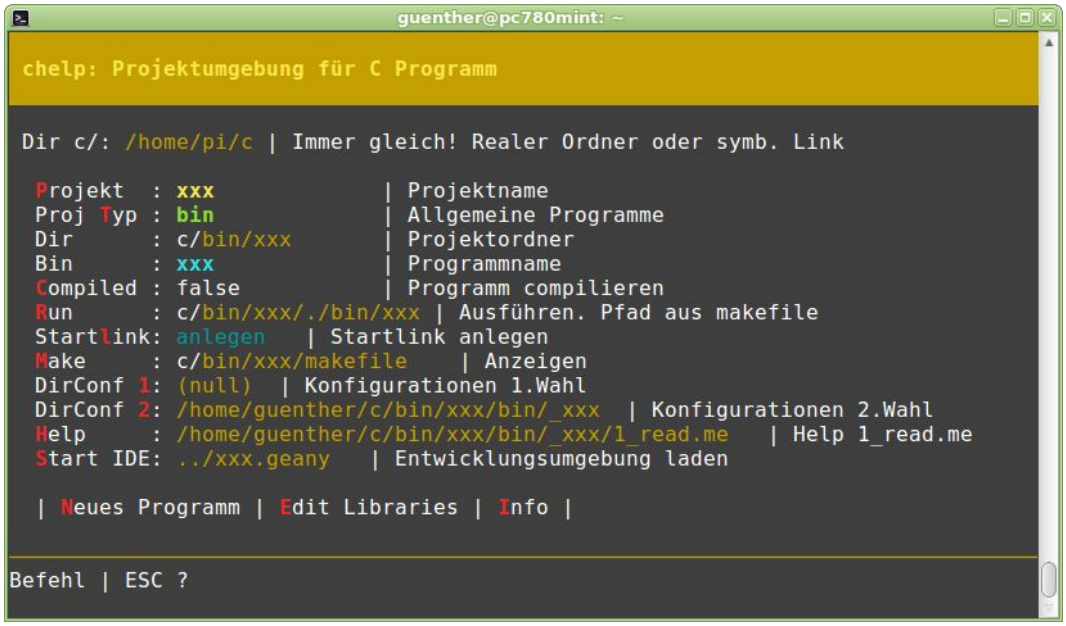
Neues Projekt anlegen.
Die Durchführung:

Das neu erzeugte Programm kann mit chelp bearbeitet werden.

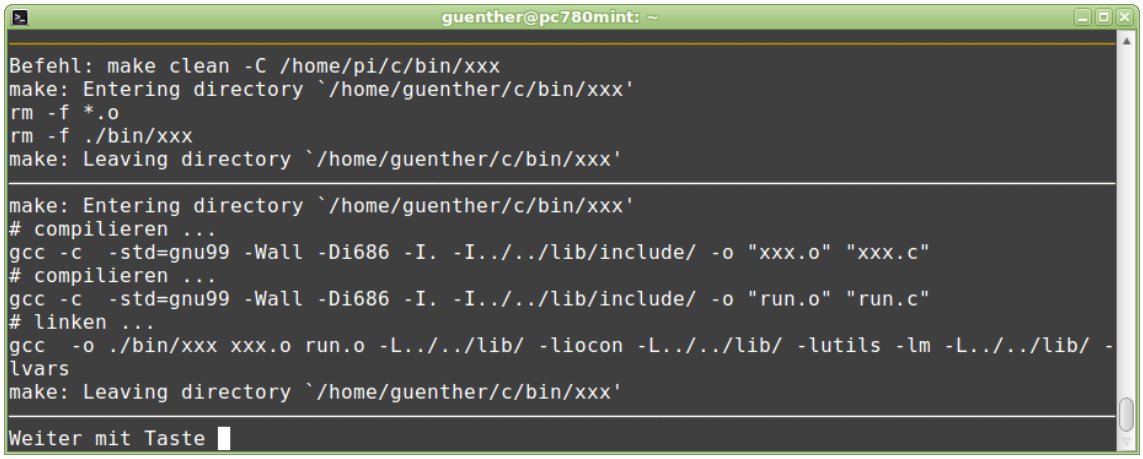
Sourcecode



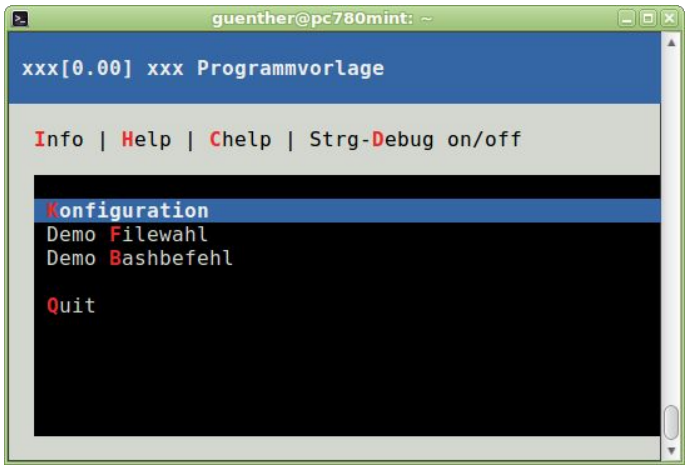
Programm mit chelp einstellen und testen:



Compilieren:
Compiled



Programm ausführen:
Run



Das Programmgerüst kann nun erweitert werden.

Programm bearbeiten:

Zur weiteren Bearbeitung des Programms xxx kann die Entwicklungsumgebung geany verwendet werden.

Befehl: Start IDE

Die Einrichtung/Verwendung von geany wird nachfolgend beschrieben.

Sourcedateien des Programms

makefile

newprg erzeugt ein passendes Standard makefile für Programm xxx

xxx.h

Main Header von Programm 'xxx'

```
#ifndef xxx_H
#define xxx_H

#define _GNU_SOURCE     // Linux GNU Erweiterungen nutzen
#include <stdio.h>      // Libc Standard-Bibliotheken
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <unistd.h>
//                      // Libs für Projekt c/ | Alle Header in c/lib/include
#include "utils.h"      // libutils.a | Strings, System- und Hilfsfunktionen
#include "vars.h"       // ibvars.a   | Globale Variablen
#include "script.h"     // libvars.a  | Konfigurations- und Scriptdateien
#include "err.h"        // libutils.a | Fehlerobjekt Err
#include "iocon.h"      // libiocon.a | Input-Output-CONsole oder Terminals.
#include "files.h"      // libutils.a | Dateisystemfunktionen
#include "boxmenu.h"    // libiocon.a | Menüdialog
#include "boxdirwahl.h" // libiocon.a | Ordner-, Device-, oder Dateiwahldialog
```

```
// Testzeile Umlaute ÄÖÜ: Zum Überprüfen der UTF-8 Einstellung des Editors
```

Globale Konstanten, Variablen und Funktionen des Programms

```
// ============================================================================
// Globale Konstanten
//
#define VERSION         "0.00"           // 2020-01-18
#define READMe          "1_read.me"      // Hilfedatei
//
#define CONFIGSuffix0   ".conf"               // Suffix Konfiguration
#define CONFIGSuffix    "*"CONFIGSuffix0      // Suffix Konfiguration
#define CONFIGNameDef   "test"CONFIGSuffix0  // Default Konfiguration
//
#define EDITORDefault   "nano"                // Default Editor
```

```
// ============================================================================
// Globale Variablen
//
uint16_t Debug;
  // 0  : keine Debugausgaben
  // n>0: Level für Debugausgaben
  // Kann durch Aufrufparameter -d oder -D gesetzt werden.
```

Das Var-Objekt verwaltet globale Variablen. Der Zugriff erfolgt über Var-Bezeichner.
Beispiel: VarGetStr(PROGName) liefert den Programmnamen.

```
// ============================================================================
// Var-Objekt für globale Laufzeit-Variablen
//
  // Var-Bezeichner für Variablen aus Var-Objekt
  //
  // Gruppe Programm ------------------------------------------------------------
  // Diese Variablen nicht in der Konfigurationsdatei gespeichert
  #define PROGGrp      0              // Gruppe Programmvariablen
  #define PROGSubGrp   0              // Sub-Gruppe Programmvariablen
  #define PROGName     "ProgName"     // Programmname
  #define WORKDir      "WorkDir"      // Arbeitsverzeichnis
  #define CONFIG       "Config"       // Pfad zur aktuellen Konfiguration
  #define EDITOR       "Editor"       // Editor in use
  #define USER         "User"         // Username
```

```
  #define DEBUGPrint   "Debug"        // 0,1 oder 2
  //      ...                         //
  //
  // Gruppe Konfiguration -------------------------------------------------------
  // Diese Variablen werden aus der Konfigurationsdatei gelesen
  #define CONFGrp      1              // Gruppe für die Konfiguration
  #define CONFSubGrp   0
  #define XEDITOR      "XEditor"      // Var-Bezeichner für Editor für X
  #define CEDITOR      "CEditor"      // Editor für Console
```

```
  #define XTERMINAL    "XTerminal"    // Var-Bezeichner für Xterminal
  //      ...
```

Programmfunktionen

```
// ============================================================================
// Deklarationen von Run-Funktionen für Menüaufrufe
//
  void runDemo();           // Ein Demo-Befehl
  void runPrintReadMe();    // Hilfedatei 1_read.me anzeigen
  void runPrintConfig();    // Konfigurationsdatei anzeigen
  void runPrintInfos();     // Programminfos
  void runChelp();          // Programm Chelp starten
  void runPrintLess();      // Demo PrintLess
```

Globale Hilfsfunktionen

```
// ============================================================================
// Deklarationen globaler Hilfs-Funktionen
```

```
  bool callSystemChk(const char *Path);
  // Programm aufrufen mit Fehlercheck
```

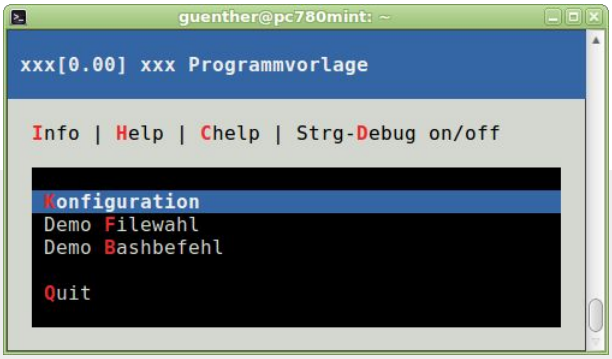
```
  void printText(const char*TextPfad, const char*Caption, const char *Farbe);
  // Textdatei anzeigen
  ...
```

```
  void Exit();                  // Exitfunktion. Aufräumen am Programmende
  void ExitSignal(int Signal);  // Abbruch durch Signal. Ruft Exit()

#endif // xxx_H Ende
```

xxx.c

Main Source

```
// Projekt c/ Programm xxx
#include <signal.h>
#include "xxx.h"  // Header xxx

uint16_t Debug = 0;  // Defaultwert: no Debug
```



Haupmenu: Das nebenstehende Menu wird durch folgende C struc Definition festgelegt.

```
// ===================================================================
// Definition des Hauptmenus.
// Die vollständige Beschreibung findet man in c/lib/include/boxmenu.h
//
tBoxMenuDef MenuMain=
{ .x=1, .y=1, .h=15, .b=0,             // Menuposition y/x, Höhe/Breite

  .Titel=                              // Titel
  "\n"
  " xxx[" VERSION "] xxx Programmvorlage\n",

  .Info=                               // Infozeile mit globalen Funktionstasten
  "\n"
  "  "FK"Info | "FK"Help | "FK"Chelp | Strg-"FK"Debug on/off\n",    // FK definiert die Farbe einer Funktionstaste

  .I=1,                                // Option: Startposition des Menu-Cursors

  .FarbeZeile =FarbeGrayBlack,       // Option: Farbe der Menuzeilen
  .FarbeCursor=FarbeWhiteBlueB,      // Option: Farbe des Menucursors


  .Items=(tBoxMenuItem [])           // Liste der Menüeinträge. Key='x' definiert die Funktionstaste.
  { {.Typ=Leer },
    {.Txt=" "FK"Konfiguration",   .Typ=Run,  .Ptr=runPrintConfig, .Key='k'},
    {.Txt=" Demo "FK"Filewahl",   .Typ=Run,  .Ptr=runDemo,        .Key='f'},
    {.Txt=" Demo "FK"Bashbefehl", .Typ=Bash, .Ptr="ls -l | less -R -P ' Weiter mit q '", .Key='b'},   // Bash Befehl
    {.Typ=Leer },
    {.Txt=" "FK"PrintLess",       .Typ=Run,  .Ptr=runPrintLess,   .Key='p'},
    {.Typ=Leer },
    {.Txt=" "FK"Quit",            .Typ=Menu  },
    { /*Menuende! Nicht löschen*/ }
  }
};
// ===========================================================================
```

Programm initialisieren:

```
void Init()
{ // Variablen und Objekte initialisieren

  clrScr(); hideCursor();  // clrScr() löscht den Bildschirm und setzt die Abmessungen

  // Filter für Var-Objekt: Programmgruppe, SubGrp und Filter setzen
  VarSetGrpFlt(PROGGrp,PROGSubGrp,false,false);

  // Programmname und -Dir bestimmen und in Var speichern
  char *Dir, *BaseName;
  getRealDirAndBaseName( getProgPath(), &Dir, &BaseName );
  VarSetStr(PROGName, BaseName);
  VarSetStr(WORKDir,  Dir);

  // ins Programmverzeichnis wechseln
  chdir(Dir);

  // Konfigurationspfad mit findConfigPfad() bestimmen
  tVar *s=VarSetStr( CONFIG, findConfigPfad( BaseName ,CONFIGNameDef));
  if (ErrChk(Err)) ErrExit("Konfiguration nicht gefunden!\n");

  // Konfiguration lesen und Variablen in Var speichern
  printf(FG);
  if (!readConfig(VarGetpStr(s), CONFGrp, Debug)) ErrExit("Fehler readConfig()\n");

  VarSetGrpFlt(CONFGrp,CONFSubGrp,false,false);  // Var Filter: Alle Vars sichtbar

  atexit(Exit);                 // exit() ruft Exit()
  signal(SIGINT,  ExitSignal);  // Option: Aufräumen nach Strg-C
  signal(SIGTERM, ExitSignal);  // Option: Aufräumen nach kill

  if (Debug>0) WeiterMitTaste();
}
```

Funktion main

Hauptprogramm

```
int main(int argc, char *argv[])
{
  // Kommandozeilenparameter bestimmen
  uint16_t i=1;
  while (i<argc)
  { if (strstr(argv[i], "-h"))
    { printf
      ( "Aufruf: %s [OPTIONS]\n"
        "OPTIONS:\n"
         "  -h     Help\n"
         "  -m,    mtrace | Speicher überwachen\n"
         "  -d, -D Debug\n"
        ,argv[0]
      );
      exit(EXIT_SUCCESS);
    }
    if (0==strcmp(argv[i], "-m")) setMallocLog(true);  // Speicherüberwachung ein
    if (0==strcmp(argv[i], "-d")) Debug=1;             // Debugmodus
    if (0==strcmp(argv[i], "-D")) Debug=2;
    i++;
  }

  Init();  // Programm initialisieren

  while (true)                                  // Main Loop
  { uint16_t Taste = runMenu(&MenuMain, Debug); // Hauptmenu starten und Menufunktionen ausführen
                                                // Rückkehr mit globaler Funktionstaste oder taste_return
    switch (low(Taste))                         // globale Funktionstasten auswerten
    {
      case 'i': runPrintInfos();   break;
      case 'h': runPrintReadMe();  break;
      case 'c': runChelp();        break;

      case taste_ctl_d:  if (Debug>0) Debug=0; else Debug=1; break;

      case taste_return:  exit(EXIT_SUCCESS);  // Programmende. Funktion Exit ausführen
    }
  }
} // ================================================================================
```

Exit() wird beim Programmende aufgerufen:

```
void Exit()
{  printLine(0);

  // Heap freigeben: Delete Objekte ...................................
  VarDelete();                // Var-Objekt freigeben

  printf("Exit x ...\n");
  chkMallocFree();            // Standardobjekte Err, tmpStr usw. freigeben.
                              // Anleitungen zu mtrace angezeigen.
  showCursor();
  normal();                   // Textausgabe normal
} // ================================================================================


// ================================================================================
void ExitSignal(int Signal)   // Exit für Signale verwenden
{ Exit();
} // ================================================================================
```

run.c

Datei run.c enthält immer grundlegende Programmfunktionen.
Im Beispiel xxx: Konfigurationsfile mit dem Dialog BoxDirWahl bestimmen

```
void runDemo()  // Demo
{ clrScr();
  const char*Path=getFilePathDlg( VarGetStr(CONFIG), "*.bak,*.conf", BoxWahlFile, "Demo Filewahl");

  clrScr();
  printBlock("Demo Filewahl\n", NULL);
  printf
  ("\n" " Filename: %s\n" "\n"
   ,(Path) ? Path : "Filewahl abgebrochen"
  );

  printLine(0);
  WeiterMitTaste();
} // --------------------------------------------------------------------------
```

Hilfsfunktion für den Dialog BoxDirWahl.

```
const char*getStartOrdner()
{ // Ordnerdialog. Rückgabe: Home-Ordner
 return "~";
} // --------------------------------------------------------------------------
```

Aufruf für Dialog BoxDirWahl.
Alle Rückgabestrings sind temporäre Strings. Die temporären Strings werden in einem Ringspeicher angelegt. Der Ringspeicher kann gleichzeitig MaxTmpStr (z.B. 100) verschiedene Strings aufnehmen. Danach wird immer der älteste Strings/Pointer freigegeben. Der String/Pointer bleibt bestehen. Siehe utils.h

```
const char *getFilePathDlg(const char*Start, const char *Suffix, tBoxWahlRet RetTyp, const char*Caption)
{
  if (!Suffix) DirFilterDelete();                    // alle Wildcards löschen
  else         DirFilterSetWildcardsStr(Suffix, ','); // Wildcards neu setzen

  return runBoxDirWahl
  (  1,                   // Position: erste Dialog-Zeile
     1,                   // Position: erste Dialog-Spalte
    -1,                   // Höhe,   <1 vom Bildschirmende
     0,                   // Breite, <1 vom Zeilenende
    Start,                // NULL oder Startordner
    DirFilterFiles,       // NULL oder Filter: Sichtbare Ordner und Dateien
    RetTyp,               // Rückgabetyp: Files, Ordner. Siehe tBoxWahlRet
    FCap4,                // NULL Farbe Titel
    Caption,              // NULL oder Titel-Zeilen mit /n getrennt
    getStartOrdner        // NULL oder externer Ordnerdialog
  );
} // --------------------------------------------------------------------------
```

Hilfe-Text anzeigen. Der Dateipfad wird mit tmpStrF(...) temporär angelegt. tmpStrF funktioniert wie prinff(...);

```
void runPrintReadMe()
{ printText(tmpStrF("%s/%s", getDirName( VarGetStr(CONFIG)), READMe),"Help:", FCap1);
} // --------------------------------------------------------------------------
```

Konfiguration anzeigen. VarGetStr(CONFIG) liefert den String-Wert zum Bezeichner CONFIG.

```
void runPrintConfig()
{ printText( VarGetStr(CONFIG), "Konfiguration:", FCap1);
} // --------------------------------------------------------------------------
```

Textfile anzeigen.
Die Anzeige erfolgt mit dem Bash-Befehl cat. Alle Ausgaben zwischen printLessOn(...) und printLessOn(...) werden mit dem pager less angezeigt.

```
void printText(const char*TextPfad, const char*Caption, const char *Farbe)
{ clrScr();

  if (!hasAccess(TextPfad, R_OK))
  { ErrAdd(Err,TextPfad); ErrPrint(Err,"Text nicht gefunden",true); return;
  }
  printLessOn(tmpStrF("%s %s\n", Str0(Caption) ,TextPfad), Farbe, Debug);
  callSystem (tmpStrF("cat %s",TextPfad) );
  printLessOff(NULL, LESSEndText, Farbe, Debug);
} // --------------------------------------------------------------------------
```

Debuginfos anzeigen. Mit VarPrintMitFilterKurz(false) können alle Var-Variablen der gewählten Gruppe angezeigt werden.

```
void runPrintInfos()
{ clrScr();

  printLessOn("\nProgramm-Infos\n", NULL, Debug); // print-Ausgaben speichern

  printf("Version     : %s\n", VERSION );
  printf("System      : %s\n", (isPi()) ? "Raspberry Pi" : "PC" );
  printf("isPi()      : %s\n", BoolToStr(isPi()));
  printf("isXRunning(): %s\n", BoolToStr(isXRunning()));
  printf("Pid         : %i\n", getpid());
  printLn();

  printBlock("Globale Programm-Variablen\n", FCap2);
  VarSetGrpFlt(PROGGrp,PROGSubGrp,true,false);
  VarPrintMitFilterKurz(false);
  printLn();

  printBlock("Globale Config-Variablen\n", FCap2);
  VarSetGrpFlt(CONFGrp,CONFSubGrp,true,false);
  VarPrintMitFilterKurz(false);
  printLn();

  printLessOff(NULL, LESSEndText, NULL, Debug);  // print-Ausgaben mit less anzeigen

  VarSetGrpFlt(PROGGrp,PROGSubGrp,false,false); // Alle Var's sichtbar

} // --------------------------------------------------------------------------
```

Standard makefile für c/ Programme

Das Standard makefile benutzt bedingte Compilierung. Für die bedingte Compilierung werden 4 Buchstaben des 'machine hardware name' verwendet.

Abfrage am Pi liefert:

```
uname -m | cut -c1-4
armv
```

Abfrage im makefile:

```
UNAME_M = ${shell uname -m | cut -c1-4}
```

UNAME_M hat also am Pi den Wert armv. Mit dem gcc Compilerflag -D kann das Define UNAME_M gesetzt werden:

```
CFLAGS    = -Wall -std=gnu99 -D$(UNAME_M)
```

Anwendung der bedingten Compilierung in Programmen:

Sourcedatei *.c

```
  printf("Bezeichner ");
  #ifdef armv
    printf("'armv6l' definiert       -> Raspberry Pi\n");
  #else
    printf("'armv6l' nicht definiert -> Linux PC\n");
  #endif
```

Standard makefile der Programmvorlage myprog_con:

```
# ========================================================================
# Makefiles Vorlage für Projekte in c. Alle Pfadangaben sind relativ.
# Für neue Projekte die Einträge nach '#|Anpassen|' anpassen!                ▷ Das Programm newpgr erledigt diese Anpassungen!
# Achtung:
# - Zeilenumbruch mit \ aber keine Blanks am Zeilenende!                     ▷ Achtung!
# - Die Befehle in den Abschnitten Befehle: mit einem Tabulator einrücken!
# ========================================================================
#
# Bedingte Compilierung für PI und PC vorbereiten.
UNAME_M = ${shell uname -m | cut -c1-4}                                      ▷ uname -m abfragen | Raspberry Pi liefert 'armv'
# Für maschinenabhängige Compilierung wird Compilerflag -D$(UNAME_M) gesetzt.
# System Pi: 'armv' ist definiert.                                           ▷ Abfrage im Programm mit #ifdef armv ... #endif
# System PC: 'armv' ist nicht definiert.


# 1. Variablen festlegen ===============================================
#
#|Anpassen| Ordner und Programmname ||||||||||||||||||||||||||||||||||||||||||||
ZIEL_DIR  = ./bin/                                                           ▷ Programmordner relativ zum aktuellen Ordner
ZIEL      = myprog                                                           ▷ Programmname

#|Anpassen| Relativer Pfad zu den statischen Libraries ||||||||||||||||||||||||
LIBS_DIR = ../../lib/                                                        ▷ lib-Ordner relativ zum aktuellen Ordner

# Relativer Pfad zu den PC/PI Headerdateien ----------------------------------
HEADER_DIR = $(LIBS_DIR)include/                                             ▷ $(LIBS_DIR) liefert Wert von LIBS_DIR


# Namen der statischen Libraries ---------------------------------------------
LIB_UTILS = utils
LIB_IOCON = iocon
LIB_VARS  = vars


#|Anpassen| Quelldateien ||||||||||||||||||||||||||||||||||||||||||||||||||||||
# INCS    = Optional, weitere Headerdateien *.h                              ▷ für Header ohne #include Befehl in der Quelldatei
# OBJS    = Hier die Namen der Quelldateien *.o eintragen.                   ▷ Quelldatei 'name.c' mit name.o eintragen
INCS      =
OBJS      = $(ZIEL).o run.o                                                  ▷ $(ZIEL) liefert myprog
# ========================================================================



# 2. Compiler Einstellungen ============================================
# Einstellungen für den Compiler. Normalerweise keine Änderung notwendig.
# Fehler erscheinen unter '# compilieren ...'
#
CC        = gcc -c
# Compiler: gcc -c oder g++ -c nur compilieren

CFLAGS    = -std=gnu99 -Wall -D$(UNAME_M)
# -Dxx bedingte Compilierung mit define 'xx'
# -std=c99 Standard oder -std=gnu99                                          ▷ Compiler mit Standard c99 verwenden

INC_DIRS = -I.\
 -I$(HEADER_DIR)
# -Ixxx  xxx ist Pfad zu Headerdateien *.h
# -I.    aktuelles Verzeichnis

DEBUG     =
# -g für Debugger
# ========================================================================
```

Fortsetzung des makefiles auf der nächsten Seite.

```
# 3. Linker Einstellungen ============================================
# Normalerweise keine Änderung notwendig.
# Fehler erscheinen beim Compilieren unter '# linken ...'
#
LD        = gcc
# Linker:  gcc oder g++ ohne -c
LDFLAGS   =
# Linkerflags

# Pfade zum Linken der Libraries. Achtung: Reihenfolge wichtig!
# Standardpfade sind vordefiniert. Eigene Libraries brauchen zwei
# Angaben: Ordner xxx -Lxxx und Name libyyy ohne lib -lyyy
LIBS  = \
  -L$(LIBS_DIR) -l$(LIB_IOCON)\
  -L$(LIBS_DIR) -l$(LIB_UTILS) -lm\                                    ▷ -lm: headers <math.h>, <complex.h>
  -L$(LIBS_DIR) -l$(LIB_VARS)
# ====================================================================



# Befehle: Ziel duch Linken erstellen ================================
# Keine Änderung notwendig.
# Abhängig von : Objektdateien
$(ZIEL): $(OBJS)
          # linken ...
          $(LD) $(LDFLAGS) -o $(ZIEL_DIR)$@ $^ $(LIBS)


# Befehle: Objekt-Dateien compilieren ================================
# Keine Änderung notwendig.
# Abhängig von : Source- und Headerdateien
%.o : %.c $(INCS)
          # compilieren ...
          $(CC) $(DEBUG) $(CFLAGS) $(INC_DIRS) -o "$@" "$<"


# Befehle: "make clean" ==============================================
# für Ziel clean: eine Datei 'clean' nicht verwenden!
.PHONY : clean

# Aufruf von "make clean" löscht alle Objektdateien
clean :
          rm -f *.o
          rm -f $(ZIEL_DIR)$(ZIEL)
```

Speicherüberwachung

Stabile Programm dürfen keine Speicherlöcher haben. Zur Überprüfung des Speicherfreigabe kann das Programm mtrace verwendet werden.

mtrace

Wird das Programm xxx mit der Option -m gestartet so wir die Speicherreservierung und die -Freigabe protokolliert. Bei Beenden des Programms xxx wird in Funktion Exit() der Befehl chkMallocFree() aufgerufen. Dabei werden alle Standardobjekte freigegeben und im Falle der Option -m wird eine Anleitung ausgegeben:

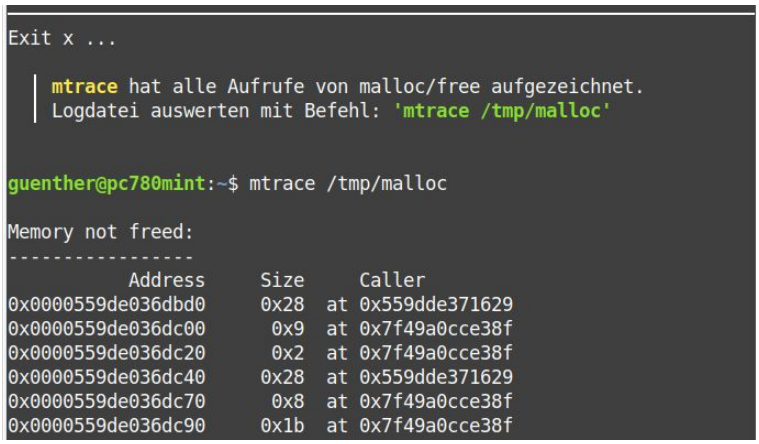
Durch Kopieren des angezeigten Befehls
export MALLOC_TRACE=/tmp/malloc
kann man die Umgebungsvariable MALLOC_TRACE setzen.

Beim nächsten Programmstart mit xxx -m wird dann die Speicherreservierung unter /tramp/allemal aufgezeichnet.

Die Prüfung erfolgt dann mit Befehl mtrace /tmp/malloc

Ok: No memory leaks

Fehlerfall: Var-Objekt wurde nicht freigegeben.



makefiles in /c

makefile für Projektordner c/

Die makefiles von Projekt c/ sind hierarchisch organisiert. Der Aufruf von make allbin in c/ compiliert dann alle Programme in c/bin.

- ▷ Der Aufruf make allbin startet das erste Makefile c/makefile.
- ▷ Befehl make -C ./bin in c/makefile startet dann Makefile bin/makefile.
- ▷ In ./bin/makefile startet der Befehl 'all :' alle Makefile's in den einzelnen Programmordnern von bin/.

Die Organisation der Makefile's kann in chelp mit dem Befehl Einstellungen > Make Dateien prüfen berprüft werden.

```
# ======================================================================
# makefile für Projektordner c/...
# Programme und Bibliotheken auf PC und Pi compilieren
#
# Make Aufrufe siehe info:
#
# Achtung: Einrückungen mit einem Tabulator! Keine Blanks am Zeilenende!
# ======================================================================
#
# Bedingte Compilierung für PI und PC:
UNAME_M = ${shell uname -m | cut -c1-4}
# Für maschinenabhängige Compilierung wird Compilerflag -D$(UNAME_M) gesetzt.
# System Pi: define armv ist definiert.
# System PC: armv ist nicht definiert.
# ======================================================================
#
info:
          # =========================================================
          # Make Aufrufe für Projekt /c auf PC und Pi
          # 2020-01-12
          # ---------------------------------------------------------
          # Im Projektordner c/
          #
          # make            // Diese Information anzeigen
          # make clean      // Alle Programme und Bibliotheken 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 mid2midraw // ALSA-MIDI 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
          # =========================================================

allbin:
          make -C ./lib
          #---------------------------------
          make -C ./bin
          #---------------------------------

all:
          make -C ./lib
          #---------------------------------
          make -C ./arduino
          #---------------------------------
          make -C ./bindemo
          #---------------------------------
          make -C ./bsp
          #---------------------------------
          make -C ./libtest
          #---------------------------------
          make -C ./bin
          #---------------------------------
#         make -C ./vorlagen
```

```
Pi:
          make -C ./lib
          #---------------------------------
          make -C ./pi
          #---------------------------------

          #=======================================
          # make all ->Ok
          #=======================================
          # Die Pi Programme aus dem Ordner c/pi/
          # verwenden GPIO.
          # Mit 'make pi' compilieren !
          #=======================================
```

```
libs:
          make -C ./lib
          #---------------------------------

chelp:
          make -C ./bin/chelp
          #---------------------------------
```

Fortsetzung auf der nächsten Seite.

```
infosys:
	make -C ./bin/infosys
	#---------------------------------
pshow:
	make -C ./bin/fbp
	make -C ./bin/pshow
	#---------------------------------

saveit:
	make -C ./bin/saveit
	#---------------------------------

mtrainer:
	make -C ./bin/mtrainer
	#---------------------------------

mid2midraw:
	make -C ./bin/mid2midraw
	#---------------------------------

kbdctl:
	make -C ./bin/kbdctl
	#---------------------------------


clock:
	# |||||||||||||||||||||||||||||||||||||||||||||||||||||||||
	# |||||||  Programme für sun900 compilieren/installieren
	# |||||||  $(HOME)/bin/clock8583 anlegen
	make -C ./pi/bin/clock8583
ifeq ($(UNAME_M), armv)
	# setuid
	# chmod u+s $(HOME)/bin/clock8583
endif


# 'make sun' für ein Projekt
sun:
	#---------------------------------
	# ||||||| $(HOME)/bin/screenstart anlegen
	make -C ./bin/screenstart
	#---------------------------------
	# ||||||| $(HOME)/bin/sun900 anlegen
	make -C ./arduino/sun_900/sun_900
	#---------------------------------


# Befehl 'make clean' löscht alle Programme und Objektfiles
#
# eine Datei 'clean' nicht als Ziel verwenden
.PHONY : clean
#
clean :
	make -C ./lib clean
	#---------------------------------
	make -C ./arduino clean
	#---------------------------------
	make -C ./bin clean
	#---------------------------------
	make -C ./bindemo clean
	#---------------------------------
	make -C ./bsp clean
	#---------------------------------
	make -C ./libtest clean
	#---------------------------------
	make -C ./pi clean
	#---------------------------------
	make -C ./vorlagen clean
	#---------------------------------

cleanMusic :
	make -C ./bin/kbdctl clean
	make -C ./bin/mtrainer clean
	#---------------------------------
```

makefile für Programmordner c/bin und c/pi/bin

Die makefiles in den Standard-Programmordnern compilieren oder löschen einzelne Programme.
Sie können mit chelp automatisch erstellt oder gelöscht werden.

Ordner c/bin:

```
# ------------------------------------------------
# makefile für alle Programme im Ordner c/bin
# Beschreibung siehe c/bin/1_read.me

all :
          make -C ./archivfoto
          make -C ./chelp
          make -C ./chksums
          make -C ./dbebau
          make -C ./fbp
          make -C ./ffvideo
          make -C ./getkey_c
          make -C ./infosys
          make -C ./kbdctl
          make -C ./lpctl
          make -C ./mid2midraw
          make -C ./midixf
          make -C ./mtrainer
          make -C ./newprg
          make -C ./pshow
          make -C ./saveit
          make -C ./screenstart
          make -C ./usbaddrun
          make -C ./x
          make -C ./xmlextract

# eine Datei 'clean' nicht als Ziel verwenden
.PHONY : clean

# Aufruf von 'make clean' löscht alle Objektfiles
clean :
          make -C ./archivfoto clean
          make -C ./chelp clean
          make -C ./chksums clean
          make -C ./dbebau clean
          make -C ./fbp clean
          make -C ./ffvideo clean
          make -C ./getkey_c clean
          make -C ./infosys clean
          make -C ./kbdctl clean
          make -C ./lpctl clean
          make -C ./mid2midraw clean
          make -C ./midixf clean
          make -C ./mtrainer clean
          make -C ./newprg clean
          make -C ./pshow clean
          make -C ./saveit clean
          make -C ./screenstart clean
          make -C ./usbaddrun clean
          make -C ./x clean
          make -C ./xmlextract clean
```

Ordner c/pi/bin:

```
# ------------------------------------------------
# makefile für alle Programme im Ordner c/pi/bin
# Beschreibung siehe c/pi/bin/1_read.me

all :
          make -C ./devtest
          make -C ./gardenctl
          make -C ./gardendown
          make -C ./picamctl
          make -C ./piclock

# eine Datei 'clean' nicht als Ziel verwenden
.PHONY : clean

# Afruf von 'make clean' löscht alle Objektfiles
clean :
          make -C ./devtest clean
          make -C ./gardenctl clean
          make -C ./gardendown clean
          make -C ./picamctl clean
          make -C ./piclock clean
```

Programmbeispiele

Allgemeine Programme

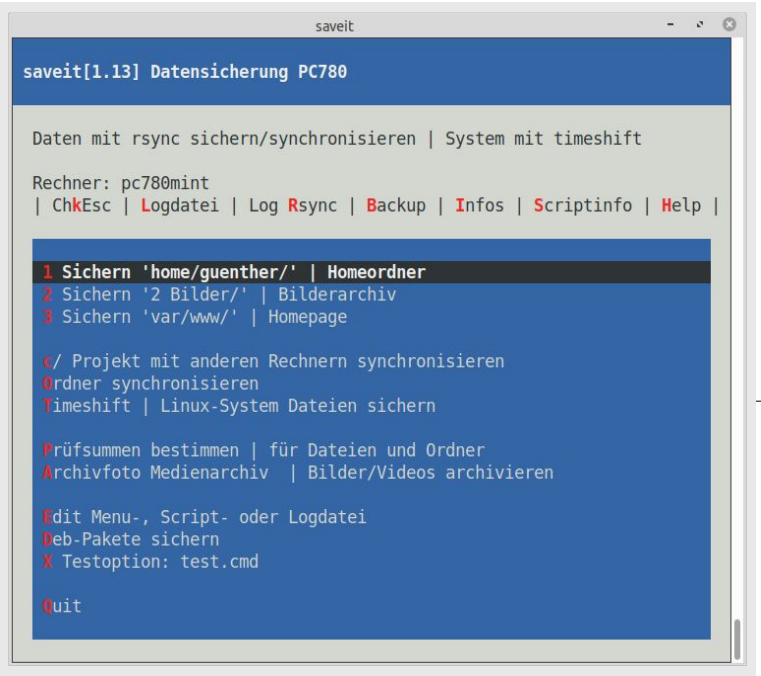
saveit

Zeigt die Verwendung von Script-Dateien.

Programm saveit bietete flexible Funktionen zur Datensicherung und zum Kopieren von Projekt /c.

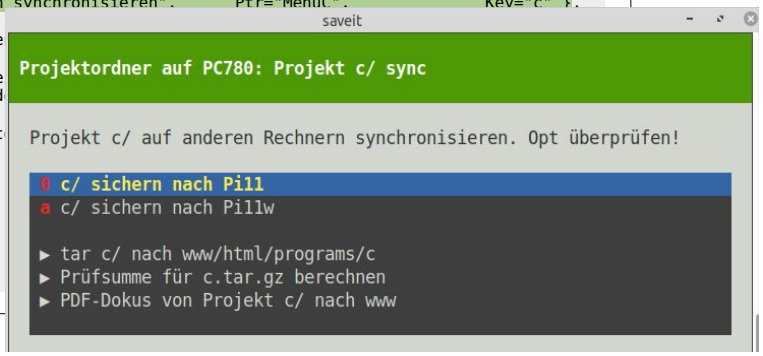
Diese Funktionen hängen vom verwendeten Rechner ab und ändern sich immer wieder.

Daher werden Menus und Sicherungs-Funktionen nicht im Programm sondern in Script-Dateien beschrieben.



Das nebenstehende Menu "MenuMain" wird durch folgendes Script definiert. DasMenu wird so wie die internen Menus auch durch struct tBoxMenuDef aus Header boxmenu.h beschrieben,

```
//=======================================================
// saveit Menüdefinitiopnen für Rechner PC780
//
// Die Menüdefinition entspricht dem Typ tBoxMenuDef aus
// Alle Parameter sind Zahlen oder Strings. Der Feldselec
// '\et' Präfix für Funktionstasten in Rot.

Menu("MenuMain",        // Hautpmenu, obligatorisch!
  y=1, x=1, h=25, b=0,  // Position y, x, Höhe, Breite

  //FarbeTitel="\e[1m\e[43m",
  //FarbeZeile="\e[1m\e[43m"
  Titel="Datensicherung PC780", // Menutitel

  Info="\n  Daten mit rsync sichern/synchronisieren | System mit timeshift\n",
  I=1,
  Items=
  {{Typ="Leer",   Txt="", Ptr="" },
   {Typ="Menu",   Txt=" \et1 Sichern 'home/guenther/' | Homeordner",          Ptr="MenuHome",          Key="1" },
   {Typ="Menu",   Txt=" \et2 Sichern '2 Bilder/' | Bilderarchiv",             Ptr="MenuBilder",        Key="2" },
   {Typ="Script", Txt=" \et3 Sichern 'var/www/' | Homepage",                  Ptr="www/www.cmd",       Key="3" },
   {Typ="Leer",   Txt="", Ptr="" },
   {Typ="Menu",   Txt=" \etc/ Projekt mit anderen Rechnern synchronisieren",  Ptr="MenuC",             Key="c" },
   {Typ="Run",    Txt=" \etOrdner synchronisieren",
   {Typ="Run",    Txt=" \etTimeshift | Linux-System Dateie
   {Typ="Leer",   Txt="", Ptr="" },
   {Typ="Run",    Txt=" \etPrüfsummen bestimmen | für Date
   {Typ="Run",    Txt=" \etArchivfoto Medienarchiv  | Bild
   {Typ="Leer",   Txt="", Ptr="" },
   {Typ="Run",    Txt=" \etEdit Menu-, Script- oder Logdat
   {Typ="Menu",   Txt=" \etDeb-Pakete sichern",
   {Typ="Script", Txt=" \etX Testoption: test.cmd",
   {Typ="Leer",   Txt="", Ptr="" },
   {Typ="Menu",   Txt=" \etQuit", Ptr="" }
  }
);
{...
```



Menu-Script "MenuC"zum nebenstehenden Menu:

```
Menu("MenuC",   // Projektordner c/ synchronisieren
  h=38,
  Titel="\n Projektordner auf PC780: Projekt c/ sync\n", // Menutitel
  FarbeTitel ="\e[1m\e[42m",            // Farbe Titel
  FarbeZeile ="\e[0;37m",
  FarbeCursor="\e[1;33m\e[44m",

  Info="\n  Projekt c/ auf anderen Rechnern synchronisieren. Opt überprüfen!\n",

  Items=
  { {Typ="Script", Txt=" \et0 c/ sichern nach Pi11",  Ptr="c_sync/C_Pi11_All.cmd",  Key="0" },
    {Typ="Script", Txt=" \eta c/ sichern nach Pi11w", Ptr="c_sync/C_Pi11w_All.cmd", Key="a" },
    {Typ="Leer",   Txt="", Ptr="" },

    {Typ="Script", Txt=" ▶ tar c/ nach www/html/programs/c",    Ptr="c_sync/WWWMakeTar.cmd" },
    {Typ="Script", Txt=" ▶ Prüfsumme für c.tar.gz berechnen",   Ptr="c_sync/WWWShasumTar.cmd" },
    {Typ="Script", Txt=" ▶ PDF-Dokus von Projekt c/ nach www",  Ptr="c_sync/WWWCopyDokus.cmd" },
    {Typ="Leer",   Txt="", Ptr="" },

    {Typ="Script", Txt=" \et1 c/ sichern nach Pi10", Ptr="c_sync/C_Pi10_All.cmd",   Key="1" },
```

Beschreibung der ersten Menu-Zeile:

```
{Typ="Script",                      Ptr ist vom Typ "Script"
 Txt=" \et0 c/ sichern nach Pi11",  Text der Menuzeile. \et0 Zeichen '0' als Funktionstaste anzeigen
 Ptr="c_sync/C_Pi11_All.cmd",       Scriptpath relativ zum Startmenu
 Key="0" },                         Funktionstaste
```

Script: "c_sync/C_Pi11_All.cmd"

```
// Befehlsdatei für saveit
// Projekt c/ auf ZielHost "pi11" speichern

ZielHost="pi11";          // Variable ZielHost setzen
ZielUser="pi";            // Variable ZielUser setzen

// Die Durchführung mit erfolgt mit einem weiteren Universal-Script
ScriptInc("c_sync/1_CNachHost.cmd"); // Script include
```

Script: c_sync/1_CNachHost.cmd

```
// Projektordner c/ auf Zielhost/ZielUser übertragen
// Scriptdatei für saveit
// Fixe Variablen: $HOST, $HOME, $Qpt1
// Variablen aus Eltern-Script: ZielHost, ZielUser

QuellDir=$HOME+"/c";
ZielDir ="/home/pi";

PrintTxt(
 "",
 " Projekt c/ mit ssh auf Raspberry " ZielHost " übertragen",
 " Quelle  : '"+QuellDir+"'",
 " ZielHost: "+ZielHost,
 " ZielUser: "+ZielUser,
 " ZielDir : "+ZielDir,
 ""
);

$Opt1     = "-av --delete --exclude='*.o' --exclude='*.a' --exclude='*.conf'";
ChkOpt1();

ZielDir= ZielUser+"@"+ZielHost+":"+ZielDir;
PrintVars("ZielHost","QuellDir","ZielDir","$Opt1");
ChkESC();

Rsync($Opt1, QuellDir, ZielDir);
ChkESC();
```

Alle Script-Variablen können entweder frei angelegt werden oder sie stammen aus Programm saveit.
Variablen aus dem Programm werden mit $ gekennzeichnet. Beispiele: $HOST, $HOME, $Qpt1.

Beispiele für Script-Funktionen:

PrintTxt()	Text anzeigen
PrintVars(...)	Werte der angegebenen Variablen anzeigen
ChkOpt1()	Check Options | Die vordefinierte Optionen prüfen oder ändern
ChkESC()	Check ESC | Abbruch prüfen
Rsync(...)	Befehl rsync aufrufen

Die Script-Variablen und Funktionen werden in Programm saveit mit dem Var-Objekt in einer fixen Scriptgruppe definiert.

Das nebenstehende Bild zeigt die Ausgaben des Befehls Infos erzeugt.

saveit

```
Scripte
Script-Objekt Info
Dir      : /home/guenther/c/bin/saveit/bin/_saveit
Grp      :  100/0     Scriptgruppe. Temp Vars
FixSubGrp:  100/1     Fixe Funktionen und Vars
BashGrp  :  100/2     Bashbefehle
BashExec : true
Debug    :    0
Ebene    :    0
Script: Var-Gruppe fixe Funktionen Variablen
Nr | Grp/Sub | Name          | Typ    | Wert
001| 100/  1 | $HOST         | Str    | "pc780mint"
002| 100/  1 | $HOME         | Str    | "/home/guenther"
003| 100/  1 | $ScriptDir    | Str    | "/home/guenther/c/bin/saveit/bin>
004| 100/  1 | $QuellDir     | Str    | "c/pi/1_Dokus"
005| 100/  1 | $ZielDir      | Str    | "/var/www/html/programs/c"
006| 100/  1 | $ArchivDir    | Str    | "/media/guenther/Linux B1"
007| 100/  1 | $Opt1         | Str    | "-av"
008| 100/  1 | $RsyncLog     | Str    | "/tmp/rsync.log"
009| 100/  1 | $NoChkESC     | Int    | 0
010| 100/  1 | ChkESC        | Ptr    | 0x55ffeb329399
011| 100/  1 | ChDir         | Ptr    | 0x55ffeb3295c0
012| 100/  1 | PrintTxt      | Ptr    | 0x55ffeb32969f
013| 100/  1 | PrintVars     | Ptr    | 0x55ffeb32972e
014| 100/  1 | ChkArchivDir  | Ptr    | 0x55ffeb328d74
015| 100/  1 | ChkOpt1       | Ptr    | 0x55ffeb32909b
016| 100/  1 | Rsync         | Ptr    | 0x55ffeb32980a
017| 100/  1 | Copy          | Ptr    | 0x55ffeb329975
018| 100/  1 | CallSystem    | Ptr    | 0x55ffeb329b9b
019| 100/  1 | SetNxtBackup  | Ptr    | 0x55ffeb329a9d
```

Scripte können auch Bash-Befehle enthalten:

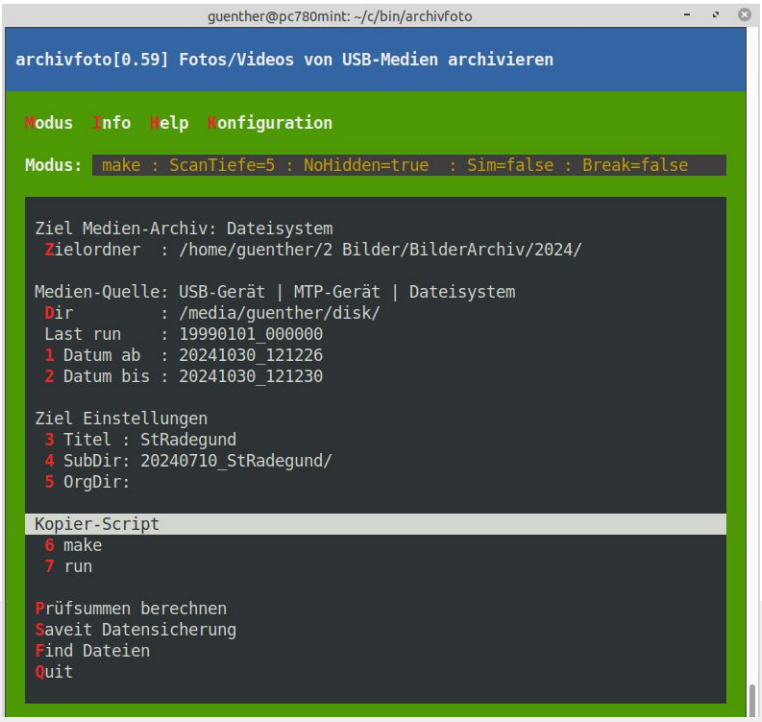
Beispiel: Projekt c/ mit tar für www komprimieren und kopieren

```
// Befehlsdatei
// Projekt c/ mit tar für www komprimieren und kopieren

$QuellDir = "c/";
$ZielDir  = "/var/www/html/programs/"$QuellDir;
$ZielName = "c.tar.gz";

PrintTxt(
 "",
 " Projekt c/ mit tar für www komprimieren und nach www kopieren",
 " Zuerst 'save' aufrufen und in einem anderen Terminal 'c/make clean'",
 " Quelle : " $HOME "/" $QuellDir,
 " Ziel   : " $ZielDir $ZielName,
 ""
 );

ChDir($HOME);
PrintVars("$QuellDir","$ZielDir","$ZielName");

Bash Prompt
// Archiv für Projekt c/ anlegen
tar
 -vcz
 --exclude="*.o" --exclude="*.a"
 -f '$ZielDir$ZielName'
 '$QuellDir'
 | less
;
```

archivfoto

Das Programm archivfoto dient zum Archivieren von Bildern und Videos
von USB-Blockgeräten und USB-MTP-Devices.

- ▷ Ermittelt die alle Medien aus den Unterverzeichnissen eines Ordners
- ▷ Die archivierten Dateinamen enthalten den Aufnahmezeitpunkt. Wenn es geht ExifTime.
- ▷ Die Archvierung erfolgt mit einem Kopier-Script. Änderungen immer möglich
- ▷ Der letzte Archvierung-Zeitpunkt wird auf dem USB-Device gespeichert
- ▷ Suchfunktionen für die Medien im Archiv



Das erzeugte Kopier-Script:

```
// Script zur Archivierung von Medien auf Datenträgern.
// Das Script wurde von 'archivfoto' erzeugt und kann
geändert werden!

// Quelle: Es werden alle Medien rekursiv kopiert.
   QuellDir ="/media/guenther/disk/";

// Ziele:
   ZielDir   ="/home/guenther/2 Bilder/BilderArchiv/2024/";  // Basis
   ZielSubDir="20240710_StRadegund/"; // Optionales: SubDir
   ZielOrgDir=""; // Optionales: OrgDir

   Ziel ="/home/guenther/2 Bilder/BilderArchiv/2024/20240710_StRadegund/";    // Zielordner für Originale

// Option: Exif | Als Speicherdatum für Fotos wird die ExifTime verwendet.

// Script-Befehle:
mkdir( Ziel ); // Zielordner anlegen

Titel ="StRadegund";    // Titel

cd( QuellDir + "DCIM/10041030" );
cp( "DSC08252.JPG", Ziel+"20241030_111226_"+Titel+".jpg" );
cp( "DSC08253.JPG", Ziel+"20241030_111230_"+Titel+".jpg" );
cp( "DSC08254.JPG", Ziel+"20241030_112947_"+Titel+".jpg" );
cp( "DSC08255.JPG", Ziel+"20241030_112949_"+Titel+".jpg" );
cp( "DSC08256.JPG", Ziel+"20241030_112951_"+Titel+".jpg" );
cp( "DSC08257.JPG", Ziel+"20241030_112952_"+Titel+".jpg" );
cp( "DSC08258.JPG", Ziel+"20241030_113116_"+Titel+".jpg" );
```

Diese Kopier-Script kann für spezielle Archivierungen vor der Ausführung temporär geändert werden.

Die Zeilen cp(Quelle, Ziel) führen den eigentlichen Kopiervorgang aus. Vor jeder cp(...) Zeile können neue Variablen definiert werden oder Titel und Ziel geändert werden.

dbebau

dbebau: Datenbank für elektronische Bauelemente

Das Programm verwendet das Objekt db um Datensätze aus Dateien '*.db' zu lesen/schreiben.
Die Datensätze werden in Arraystrukturen mit C-Syntax gespeichert.

Für die externe Speicherung wird das Objekt 'Data' (data.h) verwendet.
Die interne Verabeitung verwendet einen Arraycontainer von Typ tArray.

Siehe: clar_objekte1.pdf | Seite 19

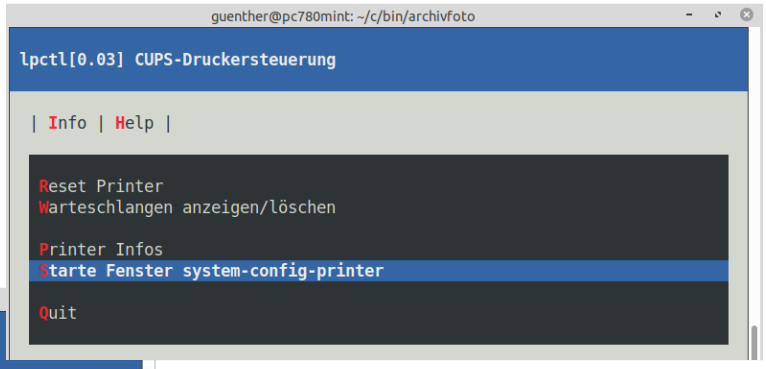
lpctl

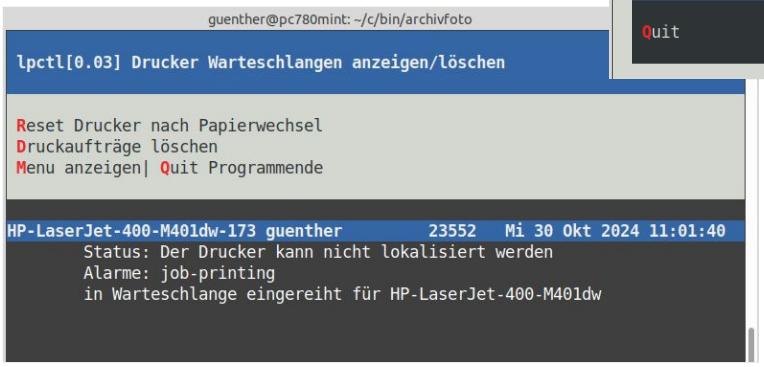
lpctl: Einfaches Hilfsprogramm für CUPS-Druckerwarteschlangen

- ▷ Druckerwarteschlangen anzeigen/löschen
- ▷ Printer Infos anzeigen
- ▷ Verwendet lpstat und clear

Beispiel: Aufruf mit mate-terminal

```
mate-terminal -e lpctl --geometry=50x17+0+0
```

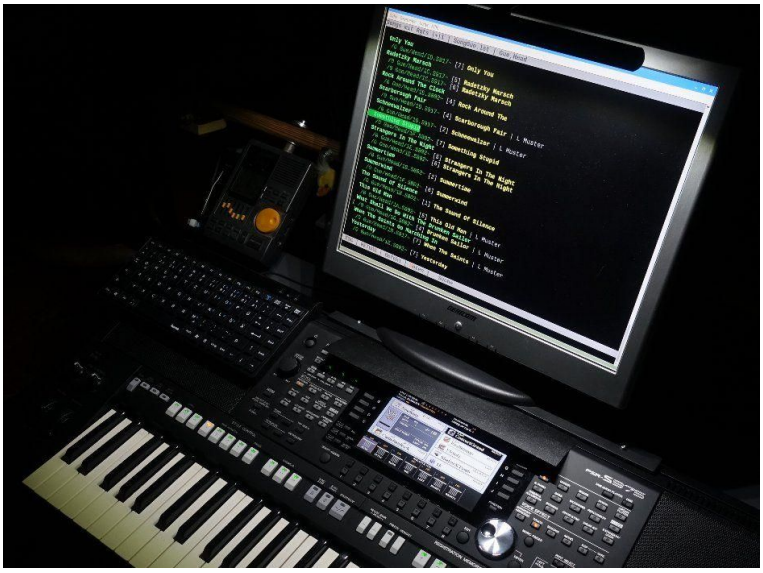




kbdctl

Songverwaltung am Raspberry Pi für Yamaha PSR-sXXX Keyboards

Doku: link_kbdctl.pdf



ffvideo

Das C-Programm 'ffvideo' kann bei der Bearbeitung von Videodateien mit ffmpeg im Terminalfenster verwendet werden. Befehle und Optionen können kommentiert in Scriptdateien zur späteren Verwendung gespeichert werden.

Doku: link_ffvideo.pdf

Raspberry PI Programme

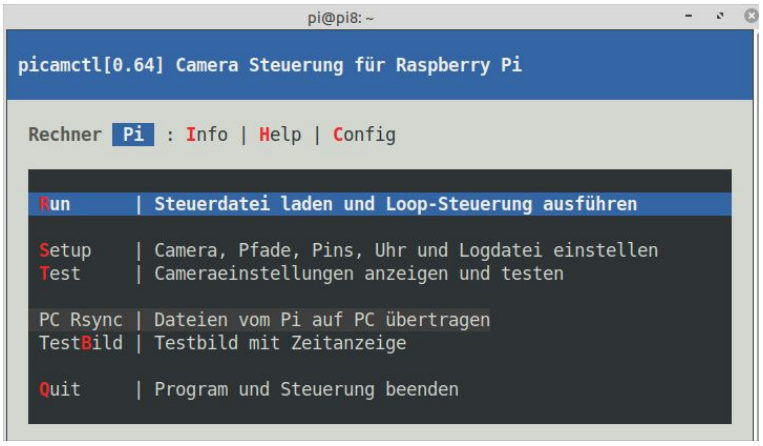
picamctl

Mit Programm picamctl können Cameras mit dem Rapberry Pi gesteuert werden.

Dokumentation: picamctl.pdf

- ▷ Camerafunktionen manuell testen
- ▷ Automatischer Betrieb mit Steuerungsdatei und/oder Bewegungssensor
- ▷ Nachtbetrieb mit Infrarotsensor
- ▷ Fernbedienung über ssh mit Terminalmultiplexer 'screen'
- ▷ Funktionen zum Übertragen der Bilder/Filme auf den PC

Für die Fernbedienung wird Programm screenstart benötigt.



screenstart

Autonom laufende Steuerprogramme für den Raspberry Pi können sehr einfach über ein Netzwerk mit ssh ferngesteuert werden.
Beim Verlassen des Terminal Remote-Terminals werden aber die gestarteten Programme gekillt. Diese Problem kann einfach mit dem Terminalumschalter screen gelöst werden.

Mit screen kann ein Programm im Hintergrund ohne Ein- und Ausgabeterminal gestartet werden. Das im Hintergrund (detached) laufende Programm kann dann von anderen Terminals aus zur Bedienung aktiviert (attached) werden.

Beim direkten Verwenden von screen kann man aber sehr leicht die Kontrolle über die Programminstanzen verlieren.

Das Hilfsprogramm screenstart ermöglicht eine einfache und stabile Verwendung von screen. Es gibt dann immer nur eine screen-Sitzung und immer nur genau eine laufende Programminstanz. Das Programm kann selbst die Aktivierung/Deaktivierug einer screen durchführen.

Beispiel: Wildcamera link_2_picamctl.pdf

Auszug aus screenstart 1_read.me:

```
 Beispiel: Kontrolle des Programms "gardenctl [opt]" mit Aufrufoptionen.

 screenstart 'gardenctl [opt]'
   | Aktiviert (attach) das Programm mit screen in einem beliebigen Terminal
   | unter dem Sitzungsnamen 'gardenctl'. Wenn die Sitzung noch nicht
   | existiert, werden alle laufenden Programm-Instanzen gekillt und
   | das Programm mit 'gardenctl [opt]' in einer screen Sitzung gestartet.

 screenstart -n 'gardenctl [opt]'
   | -n für no Attach
   | Deaktiviert (detach) das Programm. Es läuft im Hintergrund weiter.
   | Wenn die Sitzung noch nicht existiert, werden alle laufenden
   | Programm-Instanzen gekillt und das Programm mit 'gardenctl [opt]'
   | ohne Terminal in einer deaktivierten screen Sitzung gestartet.

 screenstart -i 'gardenctl'
   | -i für Infos
   | Alle Infos zum Programm und zur screen Sitzung.
   | 'gardenctl' ist der Sitzungsname. Die Programmoptionen [opt]
   | sind optional.

 screenstart -c 'gardenctl'
   | -c für nur Check
   | Überprüft Programm screen-Sitzung. Der Exitcode kann vom anderen
   | Programmen ausgewertet werden. Die Exitcodes findet man weiter unten.
   | Exit=1 ist Ok! Eine Programminstanz in screen Sitzung

 screenstart -nC 'gardenctl [opt]'
   | -C für Neustart bei Check (-c) mit Exit!=1
   | -n für no Attach
   | Dieser Aufruf kann in cron verwendet werden. Damit kann periodisch
   | geprüft werden, ob das Programm in der screen-Sitzung läuft.
   | Wenn nicht, dann wird das Programm in einer neuen screen-Sitzung
   | ohne Terminal gestartet (detached).

 screenstart -k 'gardenctl'
   | -k für kill
   | Programm und screen-Sitzung killen
 ____________________________________________________________

Aufrufe:
    screenstart [Options] [PFAD/]PROGRAMM]
    screenstart [Options] '[PFAD/]PROGRAMM [PROG_OPTIONS]'
    Weitere Optionen:
       -d  Debugmodus ein
       -h  Hilfe 1_read.me anzeigen.
       -i  Statusinformationen über PROGRAMM und screen
       -l  Logdatei schreiben.
       -k  Kill PROGRAMM und Sitzung.
       -n  No Attach. PROGRAMM detached mit screen starten.
       -C  Check mit -c. Bei EXIT_Prog1_Screen1 PROGRAMM mit screen
           neu starten.
       -c  PROGRAMM und screen checken. Ergebnis in Exit.
           EXIT_Prog1_Screen1     1  PROGRAMM und screen Ok
           EXIT_Prog1_Screen0     2  PROGRAMM Ok und kein screen
           EXIT_Prog0_Screen0     3  kein PROGRAMM und kein screen
           EXIT_NoScreenInstalled 4  screen nicht installiert
           EXIT_NoProgInstalled   5  PROGRAMM nicht installiert
```

piclock

Mit dem Aufruf piclock -s kann die Systemzeit und PCF8553 Uhrzeit ohne Menu automatisch von Steuerprogrammen synchronisiert werden.
Im Fehlerfall wird der Exitcode 1 zurückgegeben.

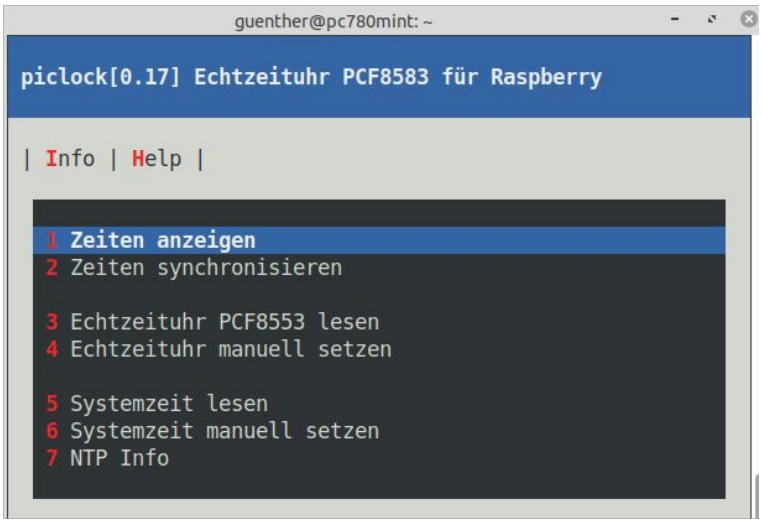
Damit steht in den Fällen Internetzugang oder Echtzeituhr eine gültige Systemzeit zur Verfügung.

Das Programm piclock dient zum Ansteuern von Echtzeituhren des Typs PCF85xx am i2c-Bus.
Anschluß an Pin I2CL SDA und Pin I2CL SCL.

Die Synchronisation von Systemzeit und PCF8553 Uhrzeit wird durch die Ausgaben des Programms timedatectl gesteuert. Diese Funktion kann mit dem Menupunkt '7 NTP Info' getestet werden.

Bei 'System clock synchronized: yes' wird die gültige Systemzeit von Pi in die Echtzeituhr PCF8583 geschrieben.

Bei 'System clock synchronized: no' wird die Zeit der Echtzeituhr PCF8583 als Systemzeit gesetzt



picamctl

Mit Programm 'picamctl' können Cameras mit dem Raspberry Pi gesteuert werden.
verschiedenen
- ▷ Steuerung von verschiedenen Cameras
- ▷ Testfunktionen für die verwendeten Pins
- ▷ Automatischer Betrieb mit einer Steuerungsdatei und/oder Bewegungssensor
- ▷ Nachtbetrieb mit Infrarotsensor
- ▷ Einfache Fernbedienung über ssh mit Terminalumschalter 'screen'
- ▷ Bilder und Filme vom Raspberry Pi mit rsync auf den PC übertragen

Siehe: c/pi/bin/picamctl/bin/_picamctl/2_picamctl.pdf

GNU General Public License

```
/*
 *
 * Copyright 2022-2024 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.
*/
```