Инструкция по установке и настройке операционной системы для контроллера

Установка

Для подготовки к компиляции, необходимо выполнить следующие шаги:

1. Распаковать prjroot.tar.gz в произвольном месте

$ tar -xvzf prjroot.tar.gz

2. Добавить необходимые переменные среды

При использовании **bash,** файлом инициализации для оболочки является **~/.bashrc**. В этот или аналогичный файл требуется добавить следующее:

```
export AS=$TARGET-as
export AR=$TARGET-ar
export LD=$TARGET-ld
export CC=$TARGET-gcc
export CPP=$TARGET-cpp
export CXX=$TARGET-g++
export RANLIB=$TARGET-ranlib
export STRIP=$TARGET-strip
export NM=$TARGET-nm
export OBJCOPY=$TARGET-objcopy
export OBJDUMP=$TARGET-objdump
export STRIP=$TARGET-strip
export STRINGS=$TARGET-strings

export PRJROOT=path-to-prjroot
export TARGET=i486-pc-linux-gnu
export HOST=i686-pc-linux-gnu
export PREFIX=$PRJROOT/tools
export SYSROOT=$PRJROOT/sysroot
export ROOTFS=$PRJROOT/rootfs
export TARGET_PREFIX=$PREFIX/$TARGET
export PATH=$PREFIX/bin:$PATH
```

2. Установить заголовочные файлы ядра

```
$ cd $PRJROOT/kernel
$ tar -xvjf linux-2.6.30.1.tar.bz2
$ cd linux-2.6.30.1
$ cp $PRJROOT/images/.config ./
$ make ARCH=x86 CROSS_COMPILE=$TARGET- menuconfig
```

В случае, когда нет необходимости вносить изменения в настройки ядра, можно сразу выйти.

$ make ARCH=x86 INSTALL_HDR_PATH=$SYSROOT/usr headers_install

3. Установить binutils

$ cd $PRJROOT/build-tools
$ tar -xvzf binutils-2.19.1.tar.gz
$ ./config-binutils
$ cd build-binutils
$ make
$ make install
$ cd ..
$ rm -r build-binutils
$ rm -r binutils-2.19.1.tar.gz

4. Установить gcc-1

$ cd $PRJROOT/build-tools
$ tar -xvzf gcc-4.4.0.tar.gz
$ tar -xvjf gmp-4.3.1.tar.bz2
$ tar -xvjf mpfr-2.4.1.tar.bz2
$ cp mpfr-2.4.1 gcc-4.4.0/mpfr
$ cp gmp-4.3.1 gcc-4.4.0/gmp
$ ./config-gcc1
$ cd build-gcc
$ make all-gcc
$ make install-gcc
$ cd ..
$ rm -r build-gcc
$ rm -r gcc-4.4.0

5. Установить eglibc-1

$ cd $PRJROOT/build-tools
$ tar -xvjf eglibc-2.10.1.tar.bz2
$ ./config-lib
$ cd build-lib
$ make install-headers install_root=$SYSROOT install-bootstrap-headers=yes
$ mkdir -p $SYSROOT/usr/lib
$ make csu/subdir_lib
$ cp csu/crt1.o csu/crti.o csu/crtn.o $SYSROOT/usr/lib
$ $TARGET-gcc -nostdlib -nostartfiles -shared -x c /dev/null -o $SYSROOT/usr/lib/libc.so

6.Установить gcc-2

$ cd $PRJROOT/build-tools
$ tar -xvzf gcc-4.4.0.tar.gz
$ tar -xvjf gmp-4.3.1.tar.bz2
$ tar -xvjf mpfr-2.4.1.tar.bz2
$ cp mpfr-2.4.1 gcc-4.4.0/mpfr
$ cp gmp-4.3.1 gcc-4.4.0/gmp

```
$ ./config-gcc2
$ cd build-gcc
$ make all-gcc all-target-libgcc
$ make install-gcc install-target-libgcc
$ cd ..
$ rm -r build-gcc
$ rm -r gcc-4.4.0
```

7. Установить eglibc-2

```
$ cd $PRJROOT/build-tools/build-lib
$ make
$ make install install_root=$SYSROOT
$ cd ..
$ rm -r build-lib
$ rm -r  eglibc-2.10.1
```

8. Установить gcc-3

```
$ cd $PRJROOT/build-tools
$ tar -xvzf gcc-4.4.0.tar.gz
$ tar -xvjf gmp-4.3.1.tar.bz2
$ tar -xvjf mpfr-2.4.1.tar.bz2
$ cp mpfr-2.4.1 gcc-4.4.0/mpfr
$ cp gmp-4.3.1 gcc-4.4.0/gmp
$ ./config-gcc3
$ cd build-gcc
$ make AS_FOR_TARGET="${TARGET}-as" LD_FOR_TARGET="${TARGET}-ld"
$ make install
$ cp -dv $TARGET_PREFIX/libgcc_s.so* $SYSROOT/lib
$ cp -dv $TARGET_PREFIX/libstdc++.so* $SYSROOT/usr/lib
$ cd ..
$ rm -r build-gcc
$ rm -r gcc-4.4.0
```

9. Установить библиотеки

```
$ cd $PRJROOT/sysapps
$ tar -xvzf zlib-1.2.3.tar.gz
$ tar -xvzf libpng-1.2.39.tar.gz
$ tar -xvzf fcgi-2.4.0.tar.gz
$ tar -xvzf freetype-2.3.9.tar.gz
$ tar -xvzf SDL-1.2.13.tar.gz
$ tar -xvzf SDL_image-1.2.7.tar.gz
$ tar -xvzf SDL_ttf-2.0.9.tar.gz

$ ./config-zlib
$ cd zlib-1.2.3
$ make
$ make install
$ cd ..
```

```
$ ./config-libpng
$ cd build-libpng
$ make
$ make install
$ cd ..

$ ./config-fastcgi
$ cd build-fastcgi
$ make
$ make install
$ cd ..

$ ./config-freetype
$ cd build-freetype
$ make
$ make install
$ cd ..

$ cd SDL-1.2.13/src/video/fbcon
$ cp -v SDL_fbevents.c{,.old}
$ sed -e "s/input\/mice/mice/g" SDL_fbevents.c.old >  SDL_fbevents.c
$ cd $PRJROOT/sysapps
$ dev/input/mice dev/mice
$ ./config-sdl
$ cd build-sdl
$ make
$ make install
$ cd ..

$ ./config-sdlimage
$ cd build-sdlimage
$ make
$ make install
$ cd ..

$ ./config-sdlttf
$ cd build-sdlttf
$ make
$ make install
$ cd ..
```

10. Компиляция кода для контроллера

Необходимый код, находящийся в **$PRJROOT/sysapps/Code**, включает в себя:

ProtocolVM

Программа, отвечающая за опрос внешних устройств и рассылку им управляющих команд.

```
$ cd $PRJROOT/sysapps/Code/ProtocolVM
$ ./build
```

$ cp protocolvm $ROOTFS/controller/protocolvm

FastCGIGUI

Программа, реализующая интерфейс между HTTP-сервером и **PrrotocolVM**, соединение с сервером происходит по протоколу FastCGI

$ cd $PRJROOT/sysapps/Code/FastCGIGUI
$ ./build
$ cp fastcgigui $ROOTFS/gui/fastcgigui

SDLGUI

Графический интерфейс, предназначенный для работы на котроллере.

$ cd $PRJROOT/sysapps/Code/SDLGUI
$ ./build
$ cp sdlgui $ROOTFS/gui/sdlgui

JSGUI

Набор скриптов для Web-интерфейса. Для установки достаточно скопировать содержимое в **$ROOTFS/www**

11. Компиляция ядра

При изменении настроек ядра следует выполнить следующие команды

$ cd $PRJROOT/kernel/linux-2.6.30.1
$ make ARCH=x86 CROSS_COMPILE=$TARGET- menuconfig
$ make ARCH=x86 CROSS_COMPILE=$TARGET-

Полученный в результате файл **arch/x86/boot/bzImage** необходимо скопировать на загрузочный диск.

Настройка

1. ProtocolVM

Список файлов протоколов, загружаемых при запуске системы, находится а файле **$ROOTFS/protocols**. Каждый из перечисленных файлов преобразуется, при отсутствии синтаксических ошибок, в последовательность команд, выполняемых интерпретатором протоколов. В случае, когда загружено несколько протоколов одновременно, интерпретатор поочередно выполняет по команде из каждого.

Протокол

Файл описания протоколов имеет следующую структуру

- Заголовок, содержащий имя и тип протокола: **protocol имя: тип**. Тип протокола может иметь значение **tcp-ip**, **com** или **socket**.
- Настройки протокола: **имя: значение**. Доступные настройки приведены в следующей таблице

Имя	Тип протокола	Тип значения	Допустимые значения	Описание
target-ip	tcp-ip	string	“ip:port”	Адрес управляемого устройства
port	com	number	-	Номер COM-порта начиная с 0
speed	com	number	-	Скорость работы COM-порта
character-size	com	number	5, 6, 7, 8	Размер символа в битах
parity	com	symbol	even, odd, none	Бит четности
file	socket	string	-	Путь к файлу типа UNIX Socket
query-period	—	number	-	Интервал между опросом в мс
timeout	—	number	-	Время ожидания ответа в мс
case-sensitive	—	symbol	true, false	Учитывать регистр при сравнении сообщений

- Описание действий при командах и запросах. Команды имеют вид **command имя аргумент1 аргумент2 … аргументN {операции}**, а запрос – **query {операции}**. Запрос выполняется автоматически с периодом не менее **query-period**, команды - при принятии соответствующего сообщения по внешнему интерфейсу. Операция может иметь вид **set имя значение**, устанавливающее указанное состояние для данного устройства в указанное значение, **send значение**, выполняющее отправку сообщения по каналу связи, соответствующему указанному типу протокола, или **receive {значение {операции} значение {операции} ... }**, выполняющее указанные действия при принятии соответствующего сообщения
- Каждый из аргументов представляет собой 32-битную переменную, значение которой определяется из строки комманды полученной через интерфейс (см. **Интерфейс**).

Сообщения, используемые операциями **send** и **receive** могут представлены в одном из следующих форматов:

- 32-битное беззнаковое число.
- Символ, соответствующий 8-битному значению. Символы **cr**, **nl**, **bs** имеют значения **0Dh**, **0Ah**, **08h**, соответственно. Специальный символ **timeout**, используемый в одной из ветвей операции **receive**, обозначает выполнение данной ветви по прошествии времени, указанного в настройках. Все остальные символы имеют значение **00h**.
- Переменная, имеющая значение 32-битного числа. Перед выполнением команды или запроса, все переменные не связаны. Связывание переменной происходит в одной из ветвей операции **receive**, в значении которой используется данная переменная. При этом переменная принимает значение участка полученного сообщения, соответствующего битам, занимаемым переменной в шаблоне. Переменная остается связанной в течение выполнения операций, содержащихся в этой ветви.
- Блок, описываемой структурой **(значение1 значение2 … значениеN)**, где каждое значение может иметь произвольный тип, кроме блок.

Для чисел и переменных может быть дополнительно указано количество бит, занимаемых значением. Подобные значения имеют форму **значение:биты**, где количество бит должно содержаться в интервале **[0, 32]**. Сообщение, размер которого не кратен 8 битам, дополняется нулями. В блоках, значения располагаются подряд, и дополнительные биты добавляются только в конце.

EBNF

Protocol = Header , Declaration , { Declaration } ;

Header = 'protocol' , Ident , ':' , Symbol ;
Declaration = Setting | Query | Command ;
Setting = Symbol , ':' , SettingValue ;

Query = 'query' , '{' , { Statement } , '}' ;
Command = 'command' , Symbol , { Variable } , '{' , { Statement } , '}' ;

Statement = 'send' , Value | 'set' , Symbol , BlockPart | 'receive' , '{' , { Branch } , '}' ;
Branch = Value , '{' , { Statement } , '}' ;

Block = '(' , { BlockPart } , ')' ;
BlockPart = Number , [':' , Number] | Variable , [':' , Number] | String | Symbol ;

Value = Block | BlockPart ;
SettingValue = Number | String | Symbol ;
Variable = Ident ;

Ident = UCLetter , { UCLetter | LCLetter } ;
Symbol = LCLetter , { '-' | LCLetter } ;

Number = Digit , { Digit } ;
Digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;

String = "" ? any character except " ? "" ;

UCLetter = 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' ;
LCLetter = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' ;

Интерфейс

Управление программой выполняется по протоколу TCP/IP через порт 4939. Доступны следующие команды:

- **list** – запрос списка устройств
- **command устройство команда аргумент1 аргумент2 … аргументN** – выполнить указанную команду для устройства с указанным именем. В случае, когда в качестве имени устройства используется **all**, команда выполняется для всех устройств. Каждый из аргументов должен быть строковым представлением беззнакового 32-битного числа. В случае когда команда принимает больше аргументов, чем переданно в сообщении, оставшиеся аргументы принимают значение 0.

Команды должны заканчиваться символом конца строки **0Ah**.
Возвращаемые описания устройств имеют вид **имя состояние значение**. Это сообщение также заканчивается символом конца строки. Значение для каждого из элементов состояния одного устройства передается в отдельной строке.

2. SDLGUI

При запуске интерфейса происходит его соединение с **ProtocolVM**.
Настройки интерфейса загружаются из файла **$ROOTFS/interface**. Описание каждого из элементов, содержащееся в файле, имеет следующую структуру:

[Тип]
параметр1 = значение1
параметр2 = значение2
...
параметрN = значениеN

Задание каждого параметра должно полностью располагаться на отдельной строке. Доступные параметры перечислены в следующей таблице:

Тип элемента	Параметр	Значение	Значение по умолчанию
Все	Name	Имя элемента. Обязательно для страниц	-
Page	Background	Путь к файлу с фоновым изображением	-
Все, кроме **Page**	Left, Top	Координаты верхнего левого угла элемента	0
Все, кроме **Page**	Pages	Список страниц для расположения элементов через пробел	-
Button, ToggleButton	Caption	Текст надписи на кнопке	-
Button, ToggleButton, Text	FontSize	Высота шрифта	32
Button, ToggleButton, Text	Color	Цвет текста: white, red, green, blue, yellow, magenta, black	black
Button, ToggleButton, Text	Font	Путь к файлу со шрифтом	-
Button, ToggleButton	UpImage, DownImage, HeldImage	Пути к файлам и изображениями кнопки в различных состояниях	-
Button	OnClick	Действие при нажатии на кнопку	-
Button	OnHold	Действие при удерживании кнопки	-
ToggleButton	OnDown, OnUp	Действия при переключении кнопки	-
Slider, ProgressBar	MinValue, MaxValue	Пределы изменения значения	0, 100
Slider, ProgressBar	Direction	Ориентация: horizontal, vertical	horizontal
Slider	Value	Исходное значение	0

Slider	BaseImage	Путь к файлу с фоном для ползунка	-
Slider	SliderUpImage, SliderDownImage	Пути к файлам с изображениями ползунка в различных состояниях	-
Slider	OnValueChange	Действие при изменении значения	-
ProgressBar	EmptyImage, FullImage	Пути к файлам с изображениеми для различных состояний элемена	
ProgressBar	Value	Источник значений	-
Indicator	Image **имя**	Пусть к файлу с изображением для указанного состояния	-
Indicator	DefaultState	Исходное состояние	-
Indicator	State	Источник состояний	-
Text	Align	Выравнивание текста: left, right, center	center
Text	Width, Height	Размер блока, в который вписывается текст	0, 0
Text	DefaultText	Исходная текстовая строка	-
Text	Text	Источник строк	-

Расположение элементов

Все элементы, кроме **Text**, располагаются на каждой из страниц, перечисленных в параметре **Pages**, с координатами указанными в параметрах и **Left** и **Top**. Элемент привязывается к прямоугольнику (**Left**, **Top**, **Width**, **Height**) в зависимости от значения параметра **Align** следующим образом:

- **left**: левый край строки совпадает с левым краем прямоугольника;
- **right**: правый край строки совпадает с правым краем прямоугольника;
- **center**: центр строки совпадает с центром прямоугольника.

Действия

Для параметров, определяющих действия при событиях, вызванных элементами, доступны два типа значений:

- **Page имя**: переключиться на страницу с указанным именем;
- **Command устройство имя аргумент1 аргумент2 … аргументN**: отправить устройству команду с указанным именем и аргументами. Если элемент имеет тип **Slider** и один из аргументов имеет значение _, вместо него подставляется текущее значение элемента.

Источники

Параметры для источников значений задаются в форме **устройство состояние**. После установки этого параметра, изменения указанного состояния устройства приведет к изменению значения элемента.

3. Web-интерфейс

Web-интерфейс загружает файл с описанием интерфейса аналогично программе **SDLGUI**. Для связи с **ProtocolVM** применяется программа **FastCGIGUI**, преобразующая HTTP-запросы в необходимую форму.