Сборка и настройка mcLinux 3.7.1 (CentOS 7 x64)

Ниже описан процесс сборки Linux 3.7.1 для платформы Multicore на инструметальной машине под управлением операционной системы CentOS 7 x64.

Переменная окружения $work, используемая ниже, должна содержать путь к корневому каталогу репозитория linuxnvcom01.

При установке операционной системы CentOS 7:

1. Создать своего пользователя, установив для него галочку "Администратор".
2. Установить компилятор gcc и библиотеку ncurses, необходимые для сборки и настройки mcLinux командой:
   sudo yum -y install gcc ncurses-devel svn

При каждом запуске CentOS 7 стартует с выключенной сетью. Её приходится каждый раз включать в панели задач (правый верхний угол - значок, изображающий сетевое подключение).

Настройка инструментов:

Порядок сборки инструментов из исходников:

1. Установить необходимые программы и библиотеки:
   sudo yum -y install gcc-c++ patch perl-devel perl-Data-Dumper.x86_64
2. перейти в директорию $work/toolchain/buildroot-for-mclinux371
3. распаковать архив buildroot-2013.11-mc371.tar.bz2, выполнив команду
   tar xvfj buildroot-2013.11-mc371.tar.bz2
4. перейти в директорию ./buildroot-2013.11/
5. убедиться, что в переменных окружения не указаны пути к инструментам сборки mipsel-buildroot-linux-uclibc-*.
6. выполнить сборку toolchain, вызвав команду make toolchain.
7. Скопировать файл specs из директории $work/tools/4host в директорию, содержащую библиотеку libgcc.a (в output/host/usr/lib/gcc/mipsel-buildroot-linux-uclibc/4.8.2).
8. Собрать программы для корневой файловой системы командой make.
9. После сборки прописать пути к собранным инструментам и библиотекам, добавив их в переменные окружения PATH и LD_LIBRARY_PATH (пути можно задать в файле .bashrc):
   export PATH=$PATH:$work/toolchain/buildroot-2013.11/output/host/usr/bin
   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$work/toolchain/buildroot-2013.11/output/host/usr/lib
10. Выйти и войти в систему (или перезагрузить компьютер).

Порядок настройки собранных инструментов:

1. распаковать архив $work/toolchain/buildroot-for-mclinux371/buildroot-2013.11-bin64-linux-mc371.tar.bz2:
   tar xvfj buildroot-2013.11-bin64-linux-mc371.tar.bz2
2. прописать пути к собранным инструментам и библиотекам, добавив их в переменные окружения PATH и LD_LIBRARY_PATH (пути можно задать в файле .bashrc):
   export PATH=$PATH:$work/toolchain/usr/bin
   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$work/toolchain/usr/lib
3. Выйти и войти в систему (или перезагрузить компьютер).

Сборка ядра:

Порядок сборки ядра следующий:

1. перейти в директорию /mclinux371/
2. создать директорию с корневой файловой системой. Для этого:
   1. Если выполнялась сборка инструментов из исходников, то скопировать папку с корневой файловой системой target из директории $work/toolchain/buildroot-2013.11/output/ в папку $work/mclinux371/target командой:
      cp -R $work/toolchain/buildroot-2013.11/output/target $work/mclinux371/target
   2. Если использовались собранные инструменты, то распаковать из архива target-mc371.tar

командой:

tar xvf target-mc371-minimal.tar

3. создать файлы устройств в корневой файловой системе (требуются права суперпользователя):

   sudo ./prepare_target.sh target (где target - имя папки с корневой файловой системой, должно совпадать с указанным в конфигурации ядра в пункте меню General_setup/Initramfs_source_file(s)) Конфигуратор ядра запускается командой ARCH=mips make menuconfig, выполненной в директории $work/mclinux371/linux-3.7.1
4. скопировать файл inittab из директории tools/4target/config/mclinux371, заменив одноимённый файл в target/etc.
5. удалить файл etc/init.d/S60nfs.
6. сконфигурировать ядро. Для примера в репозитории хранится конфигурационный файл ядра , который можно использовать в качестве основы для создания своей конфигурации. Сначала рекомендуется скопировать эти базовые настройки:

   а) перейти в папку linux-3.7.1

   б) cp ../../tools/4target/config/mclinux371/config .config
7. затем их подкорректировать настройки под свои нужды, вызвав конфигуратор ядра командой:

   ARCH=mips make menuconfig

   1. Конфигурация без сети Ethernet:
      1. в директории target/etc/init.d переименовать файл S40Network в _S40Network (в результате этого действия скрипт S40Network, запускающий сеть, работать не будет)
      2. в файле etc/inittab должна быть раскомментирована строка:

         ::sysinit:/bin/mount -o remount,rw /

         и закомментирована (или удалена) строка, монтирующая корневую файловую систему NFS (см. ниже).
      3. вызвать конфигуратор ядра командой

         ARCH=mips make menuconfig
      4. выбрать пункт меню Kernel hacking, убедиться, что выбран пункт Built-in kernel command line и отредактировать строку, расположенную под этим пунктом. Командная строка ядра должна иметь вид:

         console=ttyS0,115200N8 root=/dev/ram0
      5. выйти на верхний уровень меню и выбрать пункт General Setup, убедиться, что выбран пункт меню Initial RAM filesystem and RAM disk (initramfs/initrd) support, а в пункте Initramfs source file(s) указан правильный путь к корневой файловой системе.
   2. Конфигурация с поддержкой сети Ethernet:
      1. выполнить действия, аналогичные конфигурации без сети Ethernet, кроме пункта 1.
      2. установить требуемые ip-адрес, маску и широковещательный адрес для платы в файле etc/network/interfaces с помощью строк:

         auto eth0

         iface eth0 inet static

           address 191.167.1.20

           netmask 255.255.255.0

           gateway 191.167.1.1
      3. убедиться, что в директории target/etc/init.d есть файл S40Network (и не переименован)
   3. Конфигурация с монтированием корневой файловой системы по NFS:
      1. в файле target/etc/inittab команда перемонтирования корневой файловой системы из режима «только чтение» в режим «чтение-запись» должна выглядеть следующим образом:

         ::sysinit:/bin/mount -o remount,rw <host_ip>:</path/to/rootfs> /

         где

         <host_ip> - ip-адрес ПК, на котором расположена корневая файловая система;

         </path/to/rootfs> - полный путь к корневой файловой системе на ПК.
      2. вызвать конфигуратор ядра командой

         ARCH=mips make menuconfig
      3. выбрать пункт меню Kernel hacking, убедиться, что выбран пункт Built-in kernel command line и отредактировать строку, расположенную под этим пунктом. Командная строка ядра должна иметь вид:

         console=ttyS0,115200N8 root=/dev/nfs nfsroot=<host_ip>:</path/to/rootfs>,vers=3,nolock ip=<target_ip>:::<mask>::eth0:none:

         где

         <host_ip> - ip-адрес ПК, на котором расположена корневая файловая система;

</path/to/rootfs> - полный путь к корневой файловой системе на ПК;
<target_ip> - ip-адрес, который должен быть установлен для платы;
<mask> - маска подсети.

4. Настроить сервер NFS на ПК — см. главу ниже.

8. собрать ядро, выполнив команду
   ARCH=mips make

Собранный образ ядра с образом файловой системы будет собран под именем vmlinux в директории linux-3.7.1.

Установка mdb:

1. Перейти в директорию $work/tools/host
   cd $work/tools/host
2. Дать права на исполнение для программы mdb
   chmod +x ./mdb
3. Скопировать mdb в директорию /usr/bin
   sudo cp mdb /usr/bin
4. Проверить установку mdb (JTAG-адаптер должен быть подключен к ПК и плате, питание должно быть подано на плату):
   sudo mdb -u

В приглашении MDB дать команду reset. Результат выполнения команды должен выглядеть так:

```
mdb> reset
Device: idcode=0x40777001
OSCR:  0x200
PCfetch: 0xbfc00000
PCdec:   0
PCexec:  0
PCmem:   0
IRdec:   0
```

(Может отличаться idcode процессора).

Установка консольного терминала :

1. Установить консольный терминал:
   sudo yum -y install minicom
2. Войти в режим настройки minicom:
   sudo minicom -s
3. Выбрать пункт меню "Настройка последовательного порта" .
4. Выбрать пункт "Последовательный порт" (Нажать клавишу "A") .
5. Ввести путь к файлу устройства (как правило, /dev/ttyS* для встроенного последовательного порта и /dev/ttyUSB* для USB-COM адаптера, где * - порядковый номер устройства в системе).
6. Отключить аппаратное управление потоком (нажать клавишу "F").
7. Выйти из режима редактирование настроек последовательного порта нажатием клавиши "Ввод".
8. Выбрать пункт меню "Сохранить настройки как dfl".
9. Можно запускать программу с помощью команды:
   sudo minicom
10. Чтобы не вводить каждый раз пароль, можно добавить своего пользователя в группу dialout. После этого можно запускать программу без прав суперпользователя:
    minicom

Настройка сервера NFS:

1. Установить в системе необходимые утилиты:
   sudo yum -y install nfs-utils rpcbind.x86_64
2. Выполнить запуск и разрешение работы сервисам, связанным с NFS:
   sudo systemctl restart rpcbind
   sudo systemctl start nfs-server
   sudo systemctl start nfs-lock
   sudo systemctl start nfs-idmap
   sudo systemctl enable rpcbind

sudo systemctl enable nfs-server
sudo systemctl enable nfs-lock
sudo systemctl enable nfs-idmap

3. Отключить файрвол:
   sudo systemctl disable firewalld
   sudo systemctl stop firewalld
4. Отредактировать /etc/exports
   sudo gedit /etc/exports

   В него нужно добавить строку вида:

   /path/to/rootfs 192.168.1.190/24(rw,sync,no_subtree_check,no_root_squash,no_all_squash)

   где первое поле - путь к экспортируемой папке (той, которая будет видна извне), далее ip-адрес клиента, которому будет разрешен доступ к данной папке и маска (24 означает 255.255.255.0), далее опции подключения

5. Перезапустить сервер NFS (это действие требуется после каждой перезагрузки компьютера):
   sudo systemctl restart nfs-server

Подключиться к файловой системе NFS из mcLinux можно, даже если корневая файловая система расположена в ОЗУ, с помощью команды, выполненой на плате:

mount -t nfs -o nolock <host_ip>:</path/to/rootfs> </mount/point>

где

<host_ip> - ip-адрес ПК, на котором расположена корневая файловая система;

</path/to/rootfs> - полный путь к корневой файловой системе на ПК;

<mount/point> - точка монтирования в файловой системе mcLinux (любая пустая директория).

Отладка собранного ядра:

Для отладки собранного ядра на плате рекомендуется использовать прилагаемые gdb и gdbserver.

1) запустить gdbserver с правами администратора

sudo gdbserver --multi ip:port

- ip - адрес машины, к которой подключена отладочная плата(при локальной отладке указать 127.0.0.1)

- port - номер порта(к которому будет подключаться gdb, должен совпадать с указанным в файле gdbinit, по умолчанию используется 9000)

пример:

sudo gdbserver --multi 127.0.0.1:9000

2) отредактировать файл gdbinit

target extended-remote 127.0.0.1:9000 - изменить ip:port на те, с которыми был запущен gdbserver

set remote exec-file path\to\vmlinux - прописать путь к собранному образу ядра (если отладка на удалённой машине, то образ ядра необходомо предварительно на неё скопировать на неё)

3) запустить gdb

gdb -x /path/to/gdbinit /path/to/vmlinux

-x : использовать внешний файл инициализации (указать путь к нему)

Отладка приложений Linux с помощью GDB (в режиме командной строки) :

1. На плате запустить GDB Server с помощью команды:
   gdbserver --multi <ip-адрес платы>:9000
   9000 - это номер порта, может быть любым другим числом, не занятым другим приложением.
2. На ПК запустить GDB-клиент:
   mipsel-buildroot-linux-uclibc-gdb
3. В командной строки GDB-клиента запустить команду на подключение к серверу:
   target extended-remote <ip-адрес платы>:9000
   При этом, если соединение установлено, в окне клиента появится сообщение:
   Remote debugging using <ip-адрес платы>:9000
   А в окне сервера (на плате) появится сообщение:
   Remote debugging from host <ip-адрес ПК>
4. В клиенте текущую директорию можно узнать с помощью команды pwd. Используя команду cd можно перемещаться по файловой системе на ПК. Необходимо перейти в директорию,

содержащую отлаживаемую программу (либо запускать mipsel-buildroot-linux-uclibc-gdb в директории с отлаживаемой программой, тогда эта директория будет текущей по умолчанию) и дать команду file <имя elf-файла приложения> для загрузки таблицы символов отлаживаемой программы. Например,

```
file my_program
```

5. Указать GDB-серверу отлаживаемую программу на плате:

```
set remote exec-file my_program
```

При этом исполняемый elf-файл должен присутствовать в файловой системе mcLinux в директории, доступной через переменную окружения PATH.

6. Дать команду run для удалённого запуска приложения на плате. Рекомендуется до выполнения команды run установить хотя бы одну точку останова, например, на функцию main() с помощью команды:

```
break main
```

7. Наиболее употребимые команды GDB-клиента (режима командной строки):

break	- установка точки останова;
run	- запуск программы;
continue (c)	- продолжение выполнения программы после остановки на точке останова;
next (n)	- следующая инструкция языка Си (без входа в функцию/подпрограмму);
step (s)	- следующая инструкция языка Си (со входом в функцию/подпрограмму);
si	- следующая ассемблерная инструкция;
list (l)	- отобразить участок кода вокруг отлаживаемой строки;
print (p)	- напечатать значение переменной;
backtrace (bt)	- распечатать кадр стека;
quit	- завершить работу клиента.

Подсказку по формату вызова любой команды можно получить, набрав в строке gdb:

```
help <имя_команды>
```

Примечания:

1. Выбор UART для консоли:
   Выбор, на каком из UART будет работать консоль, осуществляется в конфигурации ядра, с помощью задания командной строки ядра (Kernel Hacking -> Default kernel command string). Если в параметре console указан файл ttyS0, то консоль будет выведена на UART0; если ttyS1 — UART1.
   Строка для UART0 должна содержать:

   ```
   console=ttyS0,115200N8
   ```

   Строка для UART1 должна содержать:

   ```
   console=ttyS1,115200N8
   ```

2. Ускорение сборки ядра:
   Сборку ядра можно существенно ускорить, особенно на современных многоядерных ПК, если использовать распараллеливание сборки, которое включается с помощью ключа -j<n> команды make, где <n> - количество параллельных нитей. Число <n> обычно задается из расчета 2 параллельные нити на 1 ядро, т. е. для 4-хядерного процессора рекомендуется команда:

   ```
   ARCH=mips make -j8
   ```