Embedding BSD

Иво Вачков

Въведение в embedded системите

- Дефиниция за embedded система според Wikipedia
  - Специално предназначение
  - Дълбока степен на интеграция между хардуер и софтуер
  - Специфични изисквания на/към средата
  - Предварително дефиниран набор от функции
- Пример:

BSD Architecture – Design Overview (1/2)

- Монолитно ядро
  - Системно зависима част
    - Архитектурна поддръжка
    - Драйвери на устройства
  - Системно независима част
    - Мениджмънт на процеси
    - Междупроцесна Комуникация
    - Файлови Системи
    - Мрежов Стек (един или няколко)
    - други
- Системна Библиотека
- Приложен Софтуер

BSD Architecture – Design Overview (2/2)

- Изводи:
  - BSD е операционна система с общо предназначение. В дизайна не могат да бъдат открити конструкции или алгоритми, предназначени за работа в embedded системи.
  - Двуслойният модел на ядрото е подходящ за дълбока интеграция между хардуер и софтуер.
  - BSD няма специфични изисквания към оперативната среда.
  - Приложните интерфейси, които предоставят ядрото и системната библиотека създават стабилен, предварително дефиниран набор от функции, на който може да се разчита.

BSD Architecture – From POST to the Shell Prompt

- BootLoader
- Kernel
- Процесът /sbin/init
- Система за runtime конфигурации (/etc/rc)
- /usr/libexec/getty
- Потребителски интерфейс
  - Текстов: /bin/login
  - Графичен: X
- Команден интерпретатор shell
- Извод: модулната организация позволява най-точно да се определи мястото на интерференцията в съществуващия дизайн.

BSD за embedded системи. Плюсове.

- Модулна архитектура на ОС
- Множество подсистеми
- Ниски изисквания към наличния хардуер
- Голям брой поддържани процесорни архитектури
- Драйвери за множество устройства
- Стандартизирано API (POSIX, SUSv3)
- Отворен код на ОС
- Множество налични приложения
- “Специализация” на BSD дистрибуциите
- Съществуват подобни проекти
- Либерален лиценз
- Наличие на специално разработени помагала

BSD за embedded системи. Минуси.

- Монолитна архитектура на ядрото
- Слаба поддръжка от страна на производителите на хардуер
- Липса на системна библиотека, подходяща за използване в embedded проекти (uClibc, dietlibc)
- Липса на специализирани build tools (CrossTools, BuildRoot, BusyBox)

Embedded Builds. Part I (1/7)

- Описание на проекта: **Изграждане на embedded BSD build, базиран на динамично свързани изпълними файлове**.
- Изисквания:
  - Операционна Система: FreeBSD 5
  - Изходен код на операционната система
  - Perl
  - Bochs / Qemu

Embedded Builds. Part I (2/7)

- Създаване на работна директория:
  - **mkdir /core**
- Създаване на файл с характеристики, възможно най-близки до външния носител. Използват се помагала като **bximage** (от пакета Bochs) или предоставеният от операционната система **dd**.
- Използва се възможността на FreeBSD да третира файлове като устройства:
  - **mdconfig -a -t vnode -f drive.img -u md0**
- Реинициализира таблицата на устройствата и се създава един FreeBSD дял, който използва максимален обем:
  - **fdisk -BI md0**

Embedded Builds. Part I (3/7)

- Записва се стандартен label и bootstrap код:
  - **bsdlabel -w -B md0s1**
- Създават се нужния на брой slices (обикновено само md0s1a):
  - **bsdlabel -e md0s1**
- Създава се файлова система върху новите slices:
  - **newfs /dev/md0s1a (newfs -b 8192 -f 1024 /dev/md0s1a)**
- Намалява се заделяното при свръх потребление място:
  - **tunefs -m 0 /dev/md0s1a**
- Монтира се така създадения псевдо диск в работната директория:
  - **mount /dev/md0s1a /core**

Embedded Builds. Part I (4/7)

- Създава се необходимата структура от директории:
  - **mtree -U -p /core -f /etc/mtree/BSD.root.dist**
  - **mtree -U -p /core/usr/ -f /etc/mtree/BSD.usr.dist**
  - **mtree -U -p /core/var -f /etc/mtree/BSD.var.dist**
- Използват се помагалата от проекта miniBSD (https://neon1.net/misc/minibsd.html):
  - **vi minibsd5.files**
  - **perl mkmini.pl minibsd5.files / /core/**
  - **perl mklibs.pl /core/ > minibsd5.libs**
  - **perl mkmini.pl minibsd5.libs / /core/**
  - **cp -p /usr/lib/pam* /core/usr/lib**
- Копира се конфигурационната директория /etc:
  - **cp -R /etc /core/etc**

Embedded Builds. Part I (5/7)

- Редактира се /core/etc/rc.conf според изискванията към embedded build-a:
  - **vi /core/etc/rc.conf**
- Създава се /core/etc/fstab със подходящо съдържание:
  - **/dev/ad0s1a / ufs ro 1 1**
  - **proc /proc procfs rw 0 0**
  - **/dev/md0 /tmp mfs rw,-s=8192 0 0**
  - **/dev/md1 /var mfs rw,-s=8192 0 0**
- Създава се файл с потребителски имена и пароли:
  - **rm /core/etc/master.passwd**
  - **pwd_mkdb -d /core/etc/ /core/etc/master.passwd**

Embedded Builds. Part I (6/7)

- Задава се парола на административния потребител:
  - **chroot /core/**
  - **passwd root**
  - **exit**
- Създава се и се копира ядрото на операционната система:
  - **cd /usr/src/sys/i386/conf**
  - **cp GENERIC mini-kernel**
  - **vi mini-kernel**
  - **cd /usr/src**
  - **make buildkernel KERNCONF=mini-kernel**
  - **make installkernel KERNCONF=mini-kernel DESTDIR=/core**

Embedded Builds. Part I (7/7)

- Заключителен етап:
  - **umount /core**
  - **mdconfig -d -u md0**
- Коментар

Embedded Builds. Part II (1/5)

- Описание на проекта: **Изграждане на embedded BSD build, базиран на статично свързани изпълними файлове.**
- Изисквания:
  - Операционна Система: NetBSD
  - Изходен код на операционната система
  - Bochs / Qemu (за тестове)

Embedded Builds. Part II (2/5)

- Подготовка на ядрото:
  - **options MEMORY_DISK_HOOKS**
  - **options MEMORY_DISK_IS_ROOT # root on ramdisk**
  - **options MEMORY_DISK_SERVER=1 # writeable ramdisk**
  - **options MEMORY_DISK_ROOT_SIZE=8192 # 4MB**
  - **options MEMORY_RBFLAGS=0 # multi user**
- Изграждане на ядрото:
  - **config mini-kernel**
  - **cp ../compile/mini-kernel**
  - **make depend; make**

Embedded Builds. Part II (3/5)

- Crunched Binaries. Config File:
  - **srcdirs /usr/src/bin /usr/src/sbin**
  - **progs init sh reboot ls**
  - **ln sh -sh**
  - **libs -lutil -lcrypt -ll -ledit -ltermcap**
- Crunched Binaries. Build Process:
  - **crunchgen -m Makefile crunch.conf**
  - **make -f Makefile objs exe**
- Подготовка на image, съдържащ файловата система:
  - **mkdir files**
  - **mkdir files/bin files/sbin files/dev**

Embedded Builds. Part II (4/5)

- Създаване на необходимите устройства в /dev:
  - **cp /dev/MAKEDEV files/dev**
  - **cd files/dev**
  - **./MAKEDEV floppy ramdisk wscons**
  - **cd ../..**
- Попълване на файловата система:
  - **cp crunch files/sbin/init**
  - **ln files/sbin/init files/bin/sh**
  - **ln files/sbin/init files/bin/ls**
  - **ln files/sbin/init files/sbin/reboot**
- Фиксиране на правата:
  - **su root**
  - **chown -R root:wheel files ; exit**

Embedded Builds. Part II (5/5)

- Създаване на image за вграждане в ядрото:
  - **makefs -s 4m -t ffs crunch.image files**
- Вграждане на image в ядрото:
  - **mdsetimage netbsd.ramdisk crunch.image**
- Заключителен етап:
  - **gzip -c netbsd.ramdisk > netbsd**
- Коментар

Заключение

Q & A