Krótka podróż w głąb systemu FreeBSD

Wojciech A. Koszek
wkoszek@FreeBSD.org

Wprowadzenie

- Krótkie wyjaśnienie czym jest FreeBSD
- Motywacja
- Założenia
- Proces portowania
- Napotkane problemy
- Podsumowanie

FreeBSD

- monolityczne, zmodularyzowane, wielowątkowe, wywłaszczalne jądro (wsparcia dla SMP)
- obecnie dostępne na architektury x86, x86-64 (AMD64, EM64T), PowerPC, UltraSPARC (wraz z Niagara), ARM, Alpha

Motywacja

- zainteresowanie architekturą komputerów
- zainteresowanie konstrukcją systemów operacyjnych (w szczególności FreeBSD) oraz programowaniem niskopoziomowym
- chęć zdobycia doświadczenia

Cel i założenia

- emulator GXemul (bardzo dobre narzędzie -- rozumie format ELF'a i nie wymaga praktycznie żadnej konfiguracji)
- doprowadzenie jądra w minimalnej konfiguracji do stanu, w którym możliwa jest poprawna kompilacja
- uniknięcie ingerencji w część źródeł niezależną od architektury komputera (*src/sys/vm*, *src/sys/kern*, *src/sys/dev/...*)
- otrzymanie działającej funkcji *printf()*

Proces portowania (I)

- toolchain pochodzący z projektu GNU dla procesorów MIPS (ten element był obecny w czasie przystąpienia do prac).
- możliwość utworzenia potrzebnych narzędzi przy pomocy celu *'kernel-toolchain'* dostępnego ***src/Makefile{.inc}***

Proces portowania (II)

- utworzenie struktury katalogów analogicznej do tej, która dostępna jest dla innych architektur (powerpc, sparc64, ia64, amd64...)
- ***src/sys/mips/mips***
  - Pliki wspólne dla różnych procesorów MIPS
- ***src/sys/mips/conf***
  - Pliki konfiguracyjne dla jądra na architekturę MIPS
- ***src/sys/mips/include***
  - Formaty danych specyficznych dla procesora MIPS

Proces portowania (III)

- tworzenie plików wypełnionych "szkieletem" struktury wywołań funkcji (na której polega część kodu jądra niezależna od architektury)
  - *src/sys/mips/mips/*
    - locore.S: magiczny symbol *_start*
    - copystr.S: *copyinstr()*, *copyoutstr()* ...
    - vm_machdep.c: *cpu_fork() ..*
    - machdep.c: *spinlock_enter()*, *spinlock_exit()*

Proces portowania (IV)

- Podłączenie wcześniej wspomnianych plików i katalogów do procesu budowania jądra:
  - ***src/sys/conf/Makefile.mips*** -- flagi przekazywane do kompilatora w czasie kompilacji jądra dla architektury MIPS
  - ***src/sys/conf/ldscript.mips*** -- sposób, w jaki linker umieści poszczególne sekcje w pliku ELF -- tutaj zdefiniowany jest magiczny symbol *_start*

(Cross-)kompilacja (1)

- Po zbudowaniu narzędzi:

```
cd src/ && make TARGET_ARCH=mips
                kernel-toolchain
```

- Kolejny etap polegał na próbie ukończenia sukcesem takiego polecenia:

```
cd src/ && make TARGET_ARCH=mips \
           NO_MODULES=yes \
           MIPS_LITTLE_ENDIAN=yes \
           buildkernel KERNCONF=MALTA
```

(Cross-)kompilacja (2)

Właściwy proces wprowadzania poprawek
(1) Kompilacja:

```
cd /usr/obj/mips/src/sys/MALTA
make NO_CLEAN=yes
```

(błąd, którego nikt się nie spodziewa)

(2) vim ..*sys/mips/<plik_z_moim_błędem*
(3) Powrót do (1)

Dodawanie wsparcia dla `printf()` (1)

- We FreeBSD oznacza dodanie wsparcia poprzez podłączenia modułu driver'a do istniejącej infrastruktury – ...*mips4k/malta/malta_cosole.c*:
  - `malta_{cnprobe, cninit, cnterm, cnputc, cngetc)`
  - `platform_start()` uruchamia `cninit()`, które „manualnie” woła wyżej wymienione funkcje
  - MIPS sprawia, że uzyskanie liter na ekranie (które jest pomocne...) jest bardzo proste...

Dodawanie wsparcia dla `printf()` (2)

- Funkcja wypisująca znak wygląda tak:

```
malta_cnputc(cp, c)
struct consdev *cp;
int c)
{
    PUTC(c);
}
```

- PUTC(c) to jedynie przypisanie wartości zmiennej (na przykład `'x'`) do określonego adresu pamięci

Podsumowanie

- Wnioski:
  - `#if 0` to najlepszy przyjaciel osoby portującej
- Ludzie związani z portem FreeBSD/mips: Oleksandr Tymoshenko (obecnie najbardziej aktywny), Wojciech A. Koszek (wkoszek@), Olivier Houchard (cognet@) (mentor w projekcie FreeBSD), Warner Losh (imp@) (wsparcie)

Koniec

Dziękuję za uwagę!

Wojciech A. Koszek
wkoszek@FreeBSD.org