Wstęp

Książka jest przeznaczona dla ludzi zawodowo lub hobbystycznie zainteresowanych zagadnieniami związanymi z Linuksem w systemach wbudowanych. Początkujący dzięki niej zapozna się z tematem, zaawansowany programista znajdzie gotowe pomysły i rozwiązania przydatne w pracy.

Dlaczego warto skorzystać z mojej propozycji?

1. Przeczytałem wiele książek na ten temat i żadna mi się nie spodobała. Opisują ogólne mechanizmy i są mocno przegadane. Po lekturze nawet wysoko ocenianych pozycji Czytelnik ma nadal problem z rozpoczęciem pracy z Linuksem w systemie wbudowanym.
2. Piszę według metody: Problem –> Analiza –> Rozwiązanie.
3. Książka zawiera dużo informacji praktycznych. Konfiguracja i kompilacja są przedstawione krok po kroku na prawdziwych, powtarzalnych przykładach.
4. Wszystkie przykłady można uruchomić w emulatorze – nie trzeba dodatkowo kupować sprzętu, żeby zacząć się uczyć!
5. W książce znajdują się opisy zasad działania mechanizmów dostępnych zarówno w jądrze, jak i w programach użytkowych. Po ich lekturze Czytelnik zdobędzie całościowy pogląd na zagadnienia pracy systemu operacyjnego, a także będzie w stanie dobrać odpowiednie narzędzia do odpowiednich zadań.

Książka zawiera ponadto trudne do znalezienia i porozrzucane po serwisach internetowych informacje o słabo udokumentowanych mechanizmach i interfejsach.

Z czym się zmagamy

Praca ze współczesnym systemem wbudowanym z Linuksem na pokładzie jest zadaniem multidyscyplinarnym. Aby robić to dobrze, trzeba połączyć wiedzę z dziedziny administracji systemem związaną z jego konfiguracją, działaniem, budową, wąskimi gardłami, analizą wydajności; programowania: znajomością wysokopoziomowych języków programowania, ale także zasadami niskopoziomowego działania architektury oraz elektroniki. W praktyce umiejętności projektowania obwodów nie są konieczne, ale niezbędne jest odczytywanie schematów oraz ogólna wiedza elektroniczna z teorii obwodów.

Im więcej rozwiązań projektuję lub im więcej oglądam, biorąc udział w jakimś projekcie na kolejnych etapach jego realizacji, tym bardziej zdaję sobie sprawę, jak wiele można osiągnąć, nie odwołując się do narzędzi programistycznych, ale łącząc ze sobą w odpowiedni sposób i konfigurując narzędzia dostępne w systemie Unix (ich aktualne wersje z Linuksa). Programista powinien znać system operacyjny Linux i narzędzia w nim dostępne na tyle, żeby nie przekombinowywać tworzonych rozwiązań. Zamiast od razu sięgać do ulubionego IDE i pisać kod, należy najpierw zastanowić się, czy przypadkiem problemu nie da się rozwiązać szybciej (i taniej) za pomocą gotowych narzędzi (może już go ktoś rozwiązał?).

Czytelnik musi sobie zdawać sprawę, że nadrzędnym celem pracy z systemem jest dostarczenie produktu. Wszystko jedno, czy jest to projekt komercyjny (który przyniesie zyski), czy hobbystyczny, który służy rozwojowi osobistemu, podnoszeniu kwalifikacji czy po prostu nauce, nadrzędnym celem jest dostarczenie działającego, dokończonego i zgodnego ze specyfikacją oprogramowania w przewidzianym czasie. W tym celu trzeba być raz programistą, innym razem elektronikiem, a jeszcze innym administratorem systemu – konieczne są informacje ze wszystkich tych dziedzin oraz znajomość narzędzi i mechanizmów. W niniejszej książce zawarłem solidne podstawy tej wiedzy.

Uwagi techniczne – jak rozpocząć pracę

Praktycznie wszystkie przytoczone komendy, przykłady i narzędzia można uruchomić na dostępnych na rynku zestawach deweloperskich, ale również na wirtualnej platformie ARM Versatile – emulowanej przez darmowy program `QEMU`. Nie trzeba mieć płytki/dodatkowo płacić, żeby rozpocząć przygodę z wbudowanym Linuksem – wystarczy ta książka!

System operacyjny

Zakładamy użycie systemu Linux jako środowiska pracy. Oczywiście można pracować w innym systemie (istnieją gotowe kompilatory dla Windows czy MacOSX), ale wykorzystanie Linuksa do kompilacji oprogrmowania i współpracy z urządzeniem wbudowanym jest wygodniejsze – na obu urządzeniach

mamy to samo środowisko, te same lub podobne narzędzia, nie musimy co chwila zmieniać stylu i specyfiki pracy.

Przytoczone komendy oraz nazwy pakietów z instalowanym oprogramowaniem przetestowano na dystrybucji Debian GNU/Linux Sqeezee (wydano w lutym 2011). Wybrano ją ze względu na dostępną liczbę gotowych do zainstalowania pakietów, łatwość ich instalacji i brak konieczności kompilowania programów działających na maszynie deweloperskiej. Do pracy można używać dowolnej, ulubionej dystrybucji, należy jednak liczyć się z ryzykiem, że jakiś program nie będzie dostępny lub będzie działał inaczej niż opisany.

Autor dołożył wszelkich starań, aby sprawdzić poprawność przytaczanych skryptów zarówno w systemach o architekturze x86_64, jak i i368. Zdarza się, że występują pewne drobe różnice w nazwach dostępnych pakietów lub ich zależnościach. Nie mają one jednak wpływu na kluczowe aspekty poruszanych zagadnień.

Zagadnienia związane z instalacją systemu Debian na stacji roboczej opisane zostały w dodatku A. Zamiast instalować system bezpośrednio na komputerze, można także korzystać z jedenej z dostępnych obecnie platform wirtualizacji. Autor zdecydował się na otwarto źródłową wersję programu VirtualBox. Możliwe jest także korzystanie ze zdalnego serwera (na uczelni, w pracy). W tym ostatnim przypadku jednak czytelnik może nie mieć uprawnień do instalacji programów.

Konta użytkowników – uprawnienia

Kompilację oprogramowania wykonuje się z konta zwykłego użytkownika (uruchamianie tych poleceń z konta roota grozi uszkodzeniem plików systemowych). Natomiast instalację oprogramowania, konfigurację sieci, edycję plików w kataologu `/etc` na stacji roboczej należy wykonywać jako administrator (`root`). Najprościej jest daną komendę poprzedzić wywołaniem `sudo`, np.:

```
sudo aptitude install vim
```

konfiguracja `sudo` – por. dodatek A.

Przytoczone skrypty oraz listingi komend zakładają, że użytkownik pracuje w systemie, w którym może przełączyć się na konto administratora za pomocą komendy `sudo` w celu wykonania pojedynczych komend. Ta możliwość jest wykorzystywana na przykład do tworzenia plików urządzeń (czynność ta może być również przeprowadzona bez konieczności korzystania z uprawnień administratora – w gotowym archiwum z obrazem systemu, ale jest to rozwiązania bardziej skompilowane).

Zalecana kolejność pracy z książką

Rozdział 1. zawiera szybkie wprowadzenie do systemu operacyjnego Linux, przede wszystkim w kontekście systemów wbudowanych opartych na architekturze ARM. Dla rozpoczynających swoją przygodę z Linuksem jest to przedstawienie zasad działania opracowanych przeszło 40 lat temu w UNIX-ie, a ma-

jących zastosowanie (i skutecznych do dzisiaj). Dla doświadczonych użytkowników – rozdział ten stanowi podsumowanie informacji.

Rozdział 2. – *Szybki start* – zaczynamy przygodę z systemami wbudowanymi – w około pół godziny (zależy od czasu kompilacji) budujemy narzędzia deweloperskie oraz kompletny system dla urządzenia, a następnie uruchamiamy go w emulatorze. Najbardziej zniecierpliwieni użytkownicy powinni rozpocząć lekturę od tego rozdziału. Dysponując urządzeniem, można następnie przystąpić do uruchomienia w nim systemu (rozdział 9).

W rozdziale 3. analizujemy budowę i funkcjonowanie fundamentu każdego systemu – toolchaina – zestawu narzędzi służących do kompilacji oprogramowania dla systemu docelowego. Decyzje podjęte na tym etapie, wybrane opcje i ustalone kompromisy określą kształt naszego projektu. Rozdział zawiera instrukcję krok po kroku budowy własnego toolchaina, z użyciem najnowszych dostępnych narzędzi.

Rozdział 4. opisuje dostępne rozwiązania automatyzyjące budowanie toolchainów oraz całych dystrybucji oraz praktyczne sposoby ich użycia (z doświadczeń autora).

W Rozdziale 5. omawiane są metody emulacji zestawów startowch oraz pamięci Flash. Emulatory są używane przez niektóre narzędzia deweloperskie oraz umożliwiają poznanie i testowanie systemu bez konieczności posiadania zestawu startowego. Są znakomitymi narzędziami wspomagającymi pracę dewelopera.

Rozdział 6. opisuje dobór odpowiedniej wersji jądra systemu, omawia sposób konfiguracji jądra oraz metody jego kompilacji.

W rozdziale 7. znajduje się ogólny zarys architektury jądra, opisane są sterowniki urządzeń i ogólny framework, z którego korzystają. Zaprezentowano sterowniki klientów USB oraz GPIO i sposoby ich wykorzystania we własnych projektach, a także informacje o zarządzaniu energią.

Rozdział 8.: Jak sobie poradzić z programowaniem jądrem – praca z kodem, wybrane cechy charakterystyczne oraz elementy programowania w jądrze. Własny moduł i przykłady typowych operacji używanych do wyjmowania i wkładania danych z/do sterowników.

Rozdział 9. – opisana jest tu druga oprócz jądra najważniejsza część systemu operacyjnego, czyli system bazowy. Co się w nim znajduje, w jaki sposób jądro dostaje się do plików i uruchamia aplikacje. Jakie są typowe sposoby przechowywania danych oraz jakie są ich wady i zalety. Omówiono tu także przykłady skryptów startowcyh i podstawowej konfiguracji systemu.

Rozdział 10. to opis instalacji systemu na urządzeniu deweloperskim (poparty praktycznymi poradami i opisami kilku zestawów). Zawiera zestaw procedur oraz porady i przykłady aplikacji uruchamianych, zanim jeszcze kontrolę nad urządzeniem przejmie Linux.

Rozdział 11. Diagnostyka jądra i oprogramowania. Jak sobie poradzić z problemami w jądrze i aplikacjach, jakich narzędzi użyć.

W rozdziale 12. znajduje się zestaw pogrupowanych tematycznie projektów i narzędzi rozszerzających system bazowy. W całości stanowią one kompletną dystrybucję Linuksa dla urządzeń wbudowanych, skrojoną do specyficznych wymagań większości projektów. Rozdział ten ma za zadanie oswoić użytkownika ze specyficznymi problemami cross-kompilacji. W tym celu użyto praktycznych, przydatnych na co dzień przykładów narzędzi i aplikacji, często połączonych w unikalnu sposób. Na końcu rozdziału omówiony jest sposób przygotowania gotowej dystrybucji dla systemu wbudowanego (wraz z kompletnym, działającym skryptem).

Dodatek A to opis instalacji i podstawowej konfiguracji systemu Debian GNU/Linux w środowisku wirtualnym i bezpośrednio na stacji roboczej.

Dodatek B zawiera zestaw najczęściej używanych komend powłoki i opisy narzędzi uruchamianych w powłoce zarówno na stacji robioczej, jak i w systemie uruchamianym na urządzeniu.

Dodatek C – praca z kodem źródłowym. Zawiera wprowadzenie do systemu kontroli wersji GIT stosowanego w wielu projektach, z którymi stykamy się w świecie opensource.

Zestaw uruchomieniowy

Posiadanie zestawu uruchomieniowego nie jest konieczne do pracy z książką. Opisywane systemy i aplikacjie działają znakomicie w emulatorze QEMU. Autor podczas pracy nad tekstem używał kilku zestawów uruchomieniowych dostępnych obecnie na rynku. Wszystkie one bazują na procesorach rodziny ARM9 i nowszych (na przykład Cortex-A8) i dysponują co najmniej 32 MB pamięci RAM.

Minimalna ilość pamięci, w której da się bezproblemowo uruchomić opisywane przykłady, to około 8 MB (łatwo sprawdzić, eksperymentując z opcjami emulatora w rozdziale 2.). Jednak ze względu na elastyczność Linuksa i możliwość jego konfiguracji na różne sposoby można również próbować uruchomić system na urządzeniach mających mniej pamięci, z tego też powodu unikam podawania jednoznacznych wartości. Warto jednak pamiętać o tym, że wewnętrzna pamięć procesora (z reguły kilkaset kilobajtów) nie wystarczy.

Dodatkowe materiały

Dodatkowe materiały i aktualizacje, a także skrypty i listingi w wygodniejszej do użycia, cyfrowej postaci znajdują się na stronie www książki:

http://bis.org.pl/systemy_wbudowane

Życzę owocnej pracy!