Opracowanie zajęć laboratoryjnych z architektury USB i urządzeń peryferyjnych

Czetwiorkina G.A., Kostiuk D.A.[1], Markina A.A.[1], Opasiak K.[2,3], Razumiejczyk V.S.[1]

[1]Brzeski Państwowy Uniwersytet Techniczny, dmitriykostiuk@gmail.com
[2]Politechnika Warszawska, k.opasiak@tele.pw.edu.pl
[3]Samsung R&D Insitute Poland, k.opasiak@samsung.com

The use of a Linux device as a configurable USB gadget is considered for the practical study of the USB bus architecture and protocols. Abstraction from low-level implementation within the OS is used to maximize focus on the features of data exchange and standard classes of devices, as well as the use of debugging and USB traffic monitoring tools. Working in user space and using the ConfigFS and FunctionFS virtual file systems simplifies learning tasks. Due to this, the study of USB-protocols, despite their much greater complexity, takes time, comparable to the study of legacy-interfaces.

Klasyczne podejście do badań programowania niskopoziomowego i architektury komputerów obejmuje stopniowe badanie procesora i odpowiadającego mu języka instrukcji, architektury systemu komputerowego i urządzeń peryferyjnych (zasad ich działania, interfejsów, połączeń i protokołów wymiany danych). Interakcja z klasycznymi (ang. legacy) interfejsami (COM, PS/2, LPT/IEEE 1284) umożliwia badanie pracy z portami I/O i przerywaniami ale w wielozadaniowych systemach operacyjnych jest to często dość trudnez zadanie. Wymaga ono napisania sterowników wykonywanych w całości lub w części w uprzywilejowanym trybie procesora i interakcji z innymi częściami systemu operacyjnego.

Dokonane wcześniej rozwiązanie tego problemu przy użyciu dedykowanych modułów jądra GNU/Linux pozwoliło nam zachować stary układ kursu na nowoczesnej i odpowiedniej platformie [1]. Klasyczne interfejsy zaczynają jednak stwarzać problemy podczas wykonywania zadań szkoleniowych, gdyż są one stopniowo zastępowane przez interfejs USB - proces ten jest całkowicie zakończony na laptopach i niemal całkowicie – na komputerach stacjonarnych.

Cechy materiału do badań

Pod względem złożoności, USB bardzo różni się od klasycznych interfejsów, nie jest to jeden prosty protokół komunikacyjny lecz cały ich stos.

Na poziomie logicznyn, urządzenie USB jest zawsze koncentratorem (ang. USB Hub) lub urządzeniem funkcjonalnym. Koncentratory są dodatkowymi punktami połączenia, a wszystkie pozostałe urządzenia dostarczają jedną lub wiele funkcjonalności (kompozytowe urządzenia USB). Każde urządzenie USB może posiadać jedną lub wiele konfiguracji. Konfiguracja jest to grupa funkcji (co najmniej jednoelementowa), które mogą być wykorzystywane w tym samym czasie. Każda funkcja ma co najmniej jeden interfejs, a interfejs zawiera zero lub

więcej punktów końcowych (ang. endpoint). Zarówno konfiguracje jak i interfejsy rozróżniane są pomiędzy sobą poprzez przypisanie im unikalnego identyfikatora od 1 do 255, przy założeniu ciągłości numeracji. Funkcja USB jest pojęciem czysto abstrakcyjnym i nie jest widoczna z perspektywy protokołu. Punkty końcowe identyfikowane są przy pomocy adresu składającego się z identyfikatora (w zakresie od 0 do 15) oraz kierunku kierunkiem transmisji danych w odpowiednim kanale (pipe). Każde urządzenie USB implementuje co najmniej jeden punkt końcowy. Posiada on adres 0 i jaki jedyny jest dwukierunkowy i dostępny natychmiast po podłączeniu urządzenia; pozostałe punkty końcowe stają się dostępne dopiero po wybraniu jednej z konfiguracji.

Interakcja z tą hierarchią odbywa się przy użyciu struktur danych o określonym formacie – deskryptorów. Każde urządzenie ma jeden deskryptor urządzenia, co najmniej jeden deskryptor konfiguracji, a ten najmniej jeden deskryptor interfejsu, który posiada do 30 deskryptorów punktów końcowych. Dodatkowo każde urządzenie może posiadać deskryptory łańcuchów znakowych w różnych językach. Wymiana danych w protokole USB jest zawsze inicjowana przez hosta. Przesyłane zestawy pakietów zależą od rodzaju transmisji jakie urządzenie oferuje na danym punkcie końcowym. Standard USB wyróżnia następujące rodzaje transjisji: masowy (ang. bulk), kontrolny (ang. control), izochroniczny (ang. isochronous) i przerwaniowy (ang. interrupt).

Interakcja z nowo podłączonym urządzeniem USB rozpoczyna sięzawsze od punktu końcowego 0, który umożliwia odczytanie deskryptorów opisujących budowę logiczną urządzenia, po czym system operacyjny wybiera jedną z dostępnych konfiguracji, a następnie wskazuje sterowniki, które będą dokonywały wymiany informacji poprzez dostępne punkty końcowe. Sterowniki przypisywane są do interfejsów ze względu na możliwość implementowania wielu niezależnych funkcji przez jedno urządzenie USB. Każdy sterownik może implementować protokół komunikacyjny zdefiniowany w jednej ze standardowych klas (na przykład USB-HID) lub protokół własny producenta urządzenia USB (ang. Vendor specific).

Opracowane zajęć laboratoryjnych

Opracowanie sterownika oraz funkcji USB jest trudnym zdaniem, nawet dla studentów, którzy potrafią już tworzyć proste moduły jądra. Zadanie komplikuje również dość złożony interfejs programistyczny (ang. API), a także różnica w implementacji oprogramowania dla hosta i urządzenia. Dlatego też, szkolenia z zakresu architektury komputerów często ograniczają się do teoretycznych badań funkcji magistrali USB bez praktycznych zadań programistycznych.

Przedstawione zadania laboratoryjne rozwiązują ten problem abstrachując implementację sterownika w systemie operacyjnym i maksymalnie przenosząc logikę pracy do przestrzeni użytkownika, aby skoncentrować się na badaniu struktury logicznej i zasad interakcji z USB.

Interakcja ze stosem USB w przestrzeni użytkownika jest dość powszechna.

Szczególnie gdy nie ma konieczności bezpośrendiej interakcji z innymi podsystemami jądra. W GNU/Linux po stronie hosta, ten poziom abstrakcji zapewnia popularna biblioteka `libusb`. Mimo najlepszy chęci autorów biblioteki, poziom skomplikowania stosu USB przekłada się również na API tej biblioteki. Wymaga ono znacznej ilości czasu na nauczenie się go, co tworzy dodatkowe obciążenie poznawcze, a zatem wydłuża również czas potrzebny na opanowanie materiału. Dlatego w proponowanym kursie szkoleniowym nie przypisuje się temu największej roli, a wzamian kładzie się nacisk na interakcję z USB po stronie urządzenia USB na poziomie wirtualnych systemów plików `ConfigFS` i `FunctionFS`.

Wirtualny system plików `ConfigFS` umożliwia tworzenie i modyfikowanie obiektów jądra z przestrzeni użytkownika. W podsystemie USB jest on używany jako interfejs do określenia logicznej budowy gadżetu USB (takie podejście jest stosowane na przykład w nowych urządzeniach z systemem Androidem). W tym przypadku tworzenie złożonej hierarchicznej struktury deskryptorów odbywa się na poziomie abstrakcji plików, tworząc/modyfikując odpowiednie pliki i katalogi [2]. Aby stworzyć w pełni dowolne urządzenia USB w przestrzeni użytkownika, `ConfigFS` jest używany w połączeniu z wirtualnym systemem plików `FunctionFS`.

Narzędzia programowe i wybór platformy

W zadaniach laboratoryjnych program lsusb służy do uzyskiwania informacji o podłączonych urządzeniach. Wykorzystywany jest także analizator pakietów Wireshark (z modułem jądra usbmon) do obserwacji komunikacji między hostem a urządzeniem.

Funkcjonalność implementowana po stronie urządzenia USB wymaga kontrolera zdolnego do działania w trybie urządzenia USB, którego nie ma w standardowych komputerach PC. Taki kontroler jest jednak często dostępny na platformach opartych na architekturze ARM. Rozsądnym wyborem są więc mikro-komputery Raspberry Pi lub podobne; w naszym przypadku była to rola zestawu płytek do badania programowania systemów wbudowanych [3] zbudowanych z wykorzystaniem procesora Exynos .

Jeśli konieczne jest uproszczenie wymagań sprzętowych poprzez wyeliminowanie interakcji z fizycznym kontrolerem USB, możliwe jest wykorzystanie modułu jądra dummy_hcd, który zapewnia jego emulację, Moduł ten tworzy w systemie parę składającą się z kontrolera urządzenia i hosta, komunikujących się wzajemnie. Z tego powodu sterownik hosta USB i sterownik gadżetu USB mogą pracować na tej samej maszynie.

Struktura zadań laboratoryjnych

Struktura zadań obejmuje pięć prac laboratoryjnych. Pierwsze zadanie pokazuje, jak wymieniane są informacje. Studenci dowiadują się o `lsusb` i `libusb` aby uzyskać *informacje o podłączonych urządzeniach USB, a także jak użyć*

analizatora pakietów `Wireshark` do śledzenia najprostszej interakcji (naciśnięcie klawisza). Drugie zadanie omawia specyfikę przesyłania danych przez USB i odpowiadającą im funkcjonalność `libusb`. Trzecie zadanie omawia kompozycję urządzenia za pomocą systemu plików `ConfigFS`. Czwarte zadanie umożliwia zaimplementowanie własnej funkcji przy użyciu systemu plików `FunctionFS`. Wreszcie celem ostatniego piątego zadania jest inżynieria wsteczna (ang. reverse-engineering) opracowanego przez firmę sterownika, która jest wykonywana przy pomocy `Wireshark`.

Referencje

1. Костюк Д.А., Жук А.М. Построение практикумов по программированию периферийных устройств и архитектуре ЭВМ на базе GNU/Linux // Тези міжнародної науково-практичної конференції FOSS LVIV–2011, Львів, 1–6 лютого 2011р. – Львів: Вид. ЛНУ ім. І. Франка, 2011. – С. 73-75.
2. Pietrasiewicz A. Make your own USB gadget // Открытые технологии: сборник материалов Тринадцатой Международной конференции разработчиков и пользователей свободного программного обеспечения Linux Vacation / Eastern Europe 2017, Гродно, 22 – 25 июня 2017 г. – С. 19–22.
3. Костюк Д.А., Кутень И.С., Разумейчик В.С. Построение практикумов по программированию встраиваемых систем // Десятая конференция «Свободное программное обеспечение в высшей школе»: тезисы докладов. Переславль, 24–25 января 2015 года. – М.: Альт Линукс, 2015. – С. 14–18.