Построение практикумов по программированию периферийных устройств и архитектуре ЭВМ на базеGNU/LINUX

Костюк Д.А., Жук А.М.

Брестский государственный технический университет, dmitriykostiuk@gmail.com

Рассматриваются возможности и особенности применения ОС Linux для изучения архитектуры, взаимодействия с периферийными устройствами и системного программирования на уровне узлов ЭВМ. Оцениваются подходы к решению данной задачи на разных уровнях абстракции, а также практические вопросы проведения соответствующих лабораторных практикумов.

Исторически практикумы по архитектуре вычислительных систем и их сопряжению с периферийным оборудованием для студентов-программистов строятся на базе 16-битной платформы Intel и ОС DOS. Переход к более современным 32- и 64-битным платформам осложнен рядом дополнительных факторов, в первую очередь повышенным уровнем абстракции от оборудования и изоляцией прикладных процессов [1]. Поскольку прикладной процесс не имеет полного доступа к аппаратным ресурсам, становится проблематичным писать простые учебные программы в стиле DOS, свободно взаимодействующие с устройствами.

GNU/Linux позволяет частично решить проблему доступа к системным ресурсам эскалацией привилегий прикладной программы для доступа к специальным механизмам, встроенным в ОС; однако для полноценного практического ознакомления с аппаратной архитектурой студенту необходимо научиться создавать драйвера устройств в виде модулей ядра. В последнем случае частью учебного процесса становится исследование исходного кода стандартных драйверов, позволяющее лучше понять архитектуру как программной, так и аппаратной подсистемы, и выгодное с точки зрения приобретения востребованной на рынке квалификации.

Взаимодействие с аппаратными узлами из пространства прикладных программ при наличии привилегий оказывается даже более простой задачей, чем в DOS. Написание драйверов для ядра Linux — также одна из наиболее простых задач в сравнении с разработкой драйверов для других современных ОС, как в отношении требуемой квалификации, так и по наличию в свободном доступе и распространенности необходимых пособий.

Традиционно при изучении архитектуры ЭВМ на базе 16-битной архитектуры программирование устройств может выполняться двумя способами: более простым, с использованием прерываний (системных вызовов) BIOS, и более сложным, через порты ввода/вывода соответствующих микроконтроллеров.

Доступ к функциям BIOS в GNU/Linux является наиболее спорным. Согласно общедоступной информации, ядро не использует их в своей работе; однако предоставляемые BIOS обработчики прерываний техниче-

ски доступны и могут использоваться при написании модулей ядра — по крайней мере, наименее разрушительные из них. Например, можно относительно безопасно использовать функции BIOS, читающие системные данные из области CMOS, или взаимодействующие с контроллером клавиатуры. Однако нецелесообразно уделять данным методам существенного внимания в курсе, поскольку либо их практическая польза ограничена, либо они небезопасны на действующей системе (как в случае функций доступа к дискам, используемых в первичном загрузчике ОС).

Альтернатива — доступ к устройствам через порты ввода/вывода — в ряде случаев оказывается не намного сложнее (как упоминавшееся выше взаимодействие с контроллером клавиатуры) и может относительно безопасно выполняться на двух уровнях абстракции:

– файл виртуального устройства /dev/port отображает пространство портов и при наличии необходимых прав доступа позволяет пользовательскому процессу выполнять чтение и запись в порты средствами файлового ввода/вывода (open, close, read, write);

– доступ к портам на уровне модуля ядра может быть выполнен классическими ассемблерными инструкциями in и out, непосредственно из ассемблерного кода модуля ядра либо из ассемблерной вставки в тексте программы на C.

Для ряда устройств, обычно не изучаемых в традиционных 16-битных практикумах, разумно воспользоваться более высокой абстракцией, предоставляемой виртуальными файловыми системами /dev/ и /proc/. Это дает оправданную экономию, позволяя хотя бы попробовать работать с теми объектами, для которых в противном случае в рамках практикума не нашлось бы учебных часов. В простейшем случае, доступ к файловой системе /proc/ позволяет ознакомиться с более полной конфигурацией оборудования (в сравнении с весьма ограниченными возможностями, предоставляемыми традиционным для классического курса анализом области CMOS). В качестве более сложного примера можно упомянуть файловый доступ к устройствам /dev/dsp и /dev/mixer (выполняемый все теми же системными вызовами read, write, а также ioctl), который при всей простоте реализации позволяет изучить принципы работы со звуковой картой и одновременно получить навыки разработки программ, анализирующих сигналы, принимаемые аудио-трактом. Аналогична ситуация с изучением взаимодействия по шине USB.

Точкой схождения двух уровней абстракции может быть написание модуля ядра, связанного с созданием собственного файла устройства в каталоге /dev/. Подобная работа проясняет, как осуществляется резервирование конкретных аппаратных ресурсов (портов, прерываний) и связь действий над ними с соответствующими операциями на уровне файловой абстракции.

Конкретный перечень задействованных в курсе системных и периферийных устройств зависит от выделенных часов и от предшествующей подготовки студентов в области системного программирования под Linux. Необходимо как минимум знание студентами особенностей работы в

консоли GNU/Linux, владение языком C и хотя бы основами ассемблера.

При организации практикума требуется учитывать и тот факт, что необходимая эскалация привилегий накладывает дополнительные ограничения на рабочую среду, в которой должны выполняться практические задания. Для запуска модулей ядра или доступа к узлам вычислительной системы из пространства пользователя студенты нуждаются в правах администратора, т.е. получают возможность легко нарушить целостность и работоспособность ОС. Проведенные эксперименты показывают [2], что решения на базе виртуализации для платформы x86 либо не обеспечивают необходимую точность эмуляции устройств в нестандартных режимах их использования (QEMU, VirtualBox), либо обладают недостаточными для комфортной работы производительностью и функционалом (эксперимент показывает точную работу эмулятора Bochs при цене 20-кратного снижения производительности по сравнению с нативным выполнением кода). Поэтому целесообразной альтернативой виртуализованным окружениям представляется использование специализированного LiveCD-подобного дистрибутива.

Литература

1. Костюк Д., Жук А. Перевод обучения программированию на ассемблере на платформу GNU/Linux // Журнал Root@UA. №1, 2009. Киев, Украина. - С. 39-41.
2. Применение платформы GNU/Linux для изучения архитектуры ЭВМ. //Современные проблемы радиотехники и телекоммуникаций «РТ — 2010» . Материалы 6-ой международной молодежной научно-технической конференции. 19 - 24 апреля 2010 г. - Севастополь, 2010. - С. 511