экзамены, поскольку демонстрирует реальный уровень профессиональной готовности специалиста.

Современные образовательные учреждения стремятся повысить конкурентоспособность выпускников, что требует новых подходов к обеспечению качества образования. Важным инструментом становится объективизация контроля знаний и умений, реализуемая через ИИС, основанную на балльно-рейтинговой модели. Она активизирует учебную деятельность, повышает ответственность студентов и формирует мотивацию к освоению новых навыков, создавая условия для комплексной оценки деятельности обучаемых и обучающих.

УДК 004.3

А.С. Наркевич, ст. преп.
(БГТУ, г. Минск)

ОСНОВЫ UEFI-ПРОГРАММИРОВАНИЯ

UEFI (от англ. Unified Extensible Firmware Interface) – унифицированный расширяемый интерфейс прошивки, обновление традиционного BIOS, поддерживающее жёсткие диски объемом более 2Тб, более быструю загрузку, расширенные функции безопасности, а также возможности для настройки графики и курсора мыши. Прошивка BIOS представляет собой набор микропрограмм для работы с аппаратурой компьютера [1].

Термин BIOS может употребляться корректно только по отношению к IBM PC-совместимым компьютерам. Для устройств, построенных на базе других архитектур, используются другие термины. Прошивка BIOS представляет набор микропрограмм, реализующих низкоуровневые API для работы с аппаратным обеспечением компьютера, а также создающих необходимую программную среду для запуска операционной системы для IBM PC-совместимых компьютеров. BIOS относится к системному программному обеспечению.

Основной проблемой традиционного BIOS являются ограничения, которые установлены для его поддержки: 16-разрядный реальный режим работы процессора с набором команд i8086, 1Мб адресуемого пространства памяти и набор периферийных устройств (клавиатура, видео адаптер, контроллер прямого доступа в память) совместимых с IBM AT. С развитием компьютерных систем в коде BIOS продолжали использоваться устаревшие технологии: прежде всего «реальный режим» работы процессора. Для замены устаревшего BIOS была предложена технология EFI.

UEFI – это новый стандарт, который пришёл на смену BIOS и поддерживается Форумом унифицированного расширенного интерфейса прошивки (Unified Extended Firmware Interface Forum).

Основные концепции, положенные в основу технологии UEFI – «минималистичность», «модульность» и поддержка разных процессоров. Прошивка UEFI может быть собрана под 32-битный или 64-битный процессоры Intel, 32- или 64-битный процессор ARM, а также для процессоров Intel Itanium. Кроме того, UEFI имеет собственный менеджер загрузки и работает с файловыми системами на диске (по умолчанию используется FAT32), а также возможно загружать драйверы для поддержки различного периферийного оборудования и любых файловых систем. Менеджер загрузки UEFI используется для выбора и загрузки операционной системы. Загрузчик ОС является приложением UEFI.

После включения компьютера UEFI, как и BIOS, выполняет первичное тестирование оборудования.

UEFI реализует свой менеджер загрузки, он поддерживает мульти-загрузку, позволяя выбирать загрузку из нескольких ОС.

С точки зрения безопасности различия между BIOS и UEFI также являются принципиальными. BIOS не содержит встроенных механизмов защиты процесса загрузки. UEFI включает поддержку механизма Secure Boot. При программировании системной памяти UEFI загрузчика, производители, дополнительно с кодом записывают сертификаты, содержащие ключи. Ключи – это пары ключей (открытый/публичный и закрытый/секретный) сформированные алгоритмом RSA. При загрузке в режиме Secure Boot UEFI проверяет подписи исполняемого кода на соответствие ключам. Если верификация не прошла – такой код не запускается.

В прошивке UEFI встроена командная оболочка (интерфейс), предназначенная для низкоуровневого управления компьютером, диагностики оборудования, обновления BIOS, управления загрузчиками и файловыми операциями до запуска операционной системы. Справочник по командам UEFI shell является встроенным и получить его можно с помощью команды help.

Архитектура и компоненты UEFI представлены на рисунке 1.

Рисунок 1 – Архитектура UEFI

На каждой фазе выполняется своя специфическая задача при старте системы: SEC – фаза безопасности; PEI – предварительная инициализация UEFI; DXE – на этой фазе загружаются и активируются драйверы устройств, создается среда выполнения и система переводится в 64-битный режим работы процессора, готовя её к загрузке ОС; BDS – фаза выбора загрузочного устройства и запуск операционной системы согласно приоритету в NVRAM; Runtime – фаза, на которой происходит загрузка операционной системы [2].

UEFI предоставляет ОС графический интерфейс для выполнения системных функций – настройки времени, управления питанием и других актуальных для пользователей задач.

UEFI позволяет запускать только приложения UEFI, работающие до запуска основной операционной системы. Это программы специального назначения, такие как загрузчики ОС, диагностические утилиты, EFI-оболочки или драйверы, управляющие оборудованием через интерфейс UEFI. Спецификация «UEFI Specification 2.10» доступна по ссылке [3].

Программирование UEFI – это разработка низкоуровневых приложений и драйверов (EFI-приложений), работающих поверх этой прошивки. EFI-приложения – это файлы с расширением .efi, которые выполняются до загрузки ОС. Для написания кода EFI-приложений используются языки ассемблера, С в 32/64-битном режиме работы процессора, возможна поддержка графики и требуемых файловых систем. Основные инструменты разработки: среда EDK II (Intel TianoCore) для сборки, для отладки часто используется эмулятор QEMU/OVMF, который позволяет тестировать код без реальной перезагрузки компьютера.

В листинге 1 представлен код простейшего UEFI-приложения. В Используется GNU-EFI – библиотека и набор инструментов для разработки приложений, работающих непосредственно в среде прошивки UEFI.

```
#include <efi.h>
#include <efilib.h>

EFI_STATUS EFI_API
efi_main(EFI_HANDLE      ImageHandle,      EFI_SYSTEM_TABLE
*SystemTable)
{
    InitializeLib(ImageHandle, SystemTable);

    Print(L"Hello, UEFI!\n");

    return EFI_SUCCESS;}
```

Листинг 1 – Код приложения «Hello, UEFI!»

Код UEFI-программы должен иметь точку входа efi_main, получать доступ к системной таблице UEFI, использовать предоставленные прошивкой сервисы для выполнения своих задач.

UEFI-программа представляет собой исполняемый файл формата PE/COFF, который запускается прошивкой UEFI и выполняется в её программной среде. В листинге 2 приведена команда компиляции.

```
gcc -ffreestanding -fshort-wchar -mno-red-zone -fno-stack-
protector \
    -I /usr/include/efi \
    -I /usr/include/efi/x86_64 \
    -I /usr/include/efi/protocol \
    -c main.c -o main.o
```

Листинг 2 – Команда компиляции

Команда компоновки представлена в листинге 3.

```
ld -shared -Bsymbolic \
   -L/usr/lib \
   -T /usr/lib/elf_x86_64_efi.lds \
   /usr/lib/crt0-efi-x86_64.o main.o \
   -o main.so -lgnuefi –lefi
```

Листинг 3 – Команда линковки

После компоновки необходимо преобразовать elf-файл в файл с расширением .efi. В листинге 4 представлена команда, выполняющая это преобразование.

```
objcopy --target=efi-app-x86_64 \
        -j .text -j .sdata -j .data -j .dynamic \
        -j .dynsym -j .rel -j .rela -j .reloc \
        main.so BOOTX64.EFI
```

Листинг 4 – Команда преобразования elf-файла в efi-файл

Запуск приложения выполняется с помощью эмулятора QEMU. Команда автоматического запуска efi-приложения представлена в листинге 5.

```
qemu-system-x86_64 \
  -bios /usr/share/ovmf/OVMF.fd \
  -drive file=fat:rw:.,format=raw
```

Листинг 5 – Команда для запуска в QEMU

В данной работе также реализовано интерактивное efi-приложение, демонстрирующее некоторые возможности среды UEFI: организация точки входа efi_main, использование системной таблицы для доступа к сервисам прошивки, взаимодействие с консолью ввода и вывода, а также работа с Boot Services [4].

Особое внимание было уделено механизму вызова функций прошивки через макрос uefi_call_wrapper, который обеспечивает кор-

ректное взаимодействие между кодом приложения и интерфейсами UEFI с учётом различий в соглашениях о вызовах.

В рамках примера была реализована простая интерактивная командная оболочка, что позволило наглядно продемонстрировать событийную модель ввода в UEFI.

Кроме того, представлен пример работы с файловой системой через протоколы Loaded Image и Simple File System, что поясняет каким образом UEFI-приложение получает доступ к устройству, с которого оно было загружено, открывает файловую систему, работает с файлами, управляет памятью и выводит данные пользователю – всё это до загрузки операционной системы [5].

Данный пример показывает, что при использовании библиотек, таких как GNU-EFI, программирование под UEFI становится достаточно понятным и доступным даже для начинающих разработчиков.

Целью работы является демонстрация использования программной среды UEFI, в которой возможно создание интерактивных приложений, работающих с входными данными пользователя, сервисами прошивки, ещё до запуска операционной системы.

ЛИТЕРАТУРА

1 Что такое UEFI, и чем он отличается от BIOS? [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/404511– Дата доступа: 04.02.2026

2 Внутренняя кухня UEFI: что это такое и как мы готовим его в YADRO [Электронный ресурс] – Режим доступа: https://habr.com/ru/companies/yadro/articles/886480/ – Дата доступа: 04.02.2026

3 UEFI Specification 2.10 [Электронный ресурс] – Режим доступа: https://uefi.org/specs/UEFI/2.10/ – Дата доступа: 04.02.2026

4 Разработка ОС [Электронный ресурс] – Режим доступа: https://mayerdev.github.io/osdev/intro/ – Дата доступа: 04.02.2026

5 Путешествие сквозь секреты прошивок: от BIOS/UEFI до OS [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/835844/– Дата доступа: 04.02.2026