Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

Индивидуальное задание

к лабораторной работе №6

Вариант 7

«Сборка ядра под целевую платформу»

Выполнила: Севостьянова Анна Викторовна

Группа: 5130904/30002

Преподаватель: Петров Александр Владимирович

Санкт-Петербург

2024г.

Оглавление

1. Введение

1.1 Актуальность

Сборка ядра Linux под целевую платформу актуальна по следующим причинам:

1. Оптимизация под аппаратное обеспечение

При сборке ядра под конкретную платформу можно оптимизировать его работу и улучшить производительность за счёт использования особенностей и оптимизаций, свойственных данному аппаратному обеспечению.

2. Поддержка новых устройств

3. Уменьшение размеров ядра

При сборке убираются неиспользуемые компоненты, что позволяет сократить размер ядра, экономить ресурсы и повышать эффективность работы системы.

4. Эффективность

Собранное оптимизированное ядро способно эксплуатировать аппаратное обеспечение наиболее эффективным образом, что влияет на стабильность и производительность системы в целом.

1.2 Цель

Целью данной работы является получение сконфигурированного и собранного ядра Linux под целевую платформу.

1.3 Задачи

1. Подготовка системы

2. Конфигурация ядра

3. Сборка ядра

2. Основная часть

В работе используется amd64, т.к. amd64 - это общее название для 64-битных микроархитектур, реализующих 64-битную микроархитектуру процессора, таких как микроархитектуры AMD Zen и Intel Core. В моем случае – Intel Core i5.

2.1 Методы решения задач

Получение исходных кодов ядра представлено в лабораторной работе №6.

1. Подготовка системы

По стандарту Debian ядро собирается с включенным форматом отладки DWARF (стандартизованный формат отладочной информации). Отключим его:

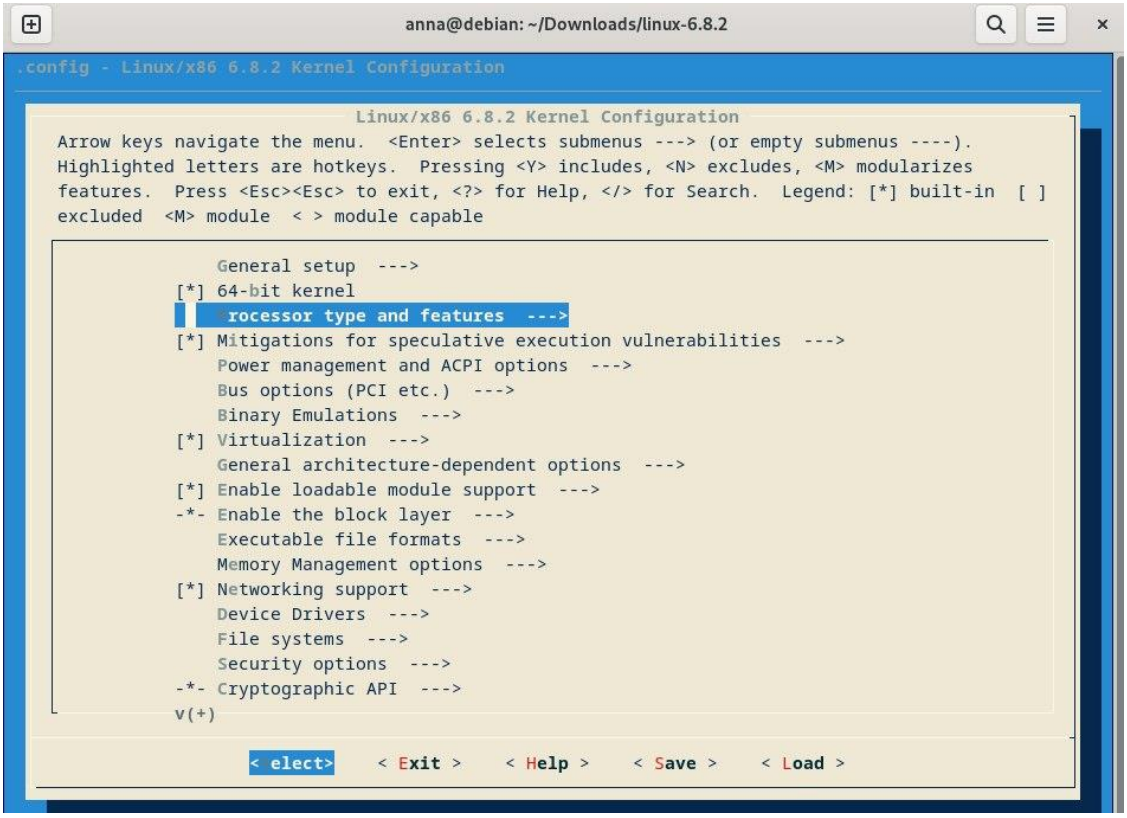
scripts/config -d CONFIG_DEBUG_INFO_DWARF_TOOLCHAIN_DEFAULT

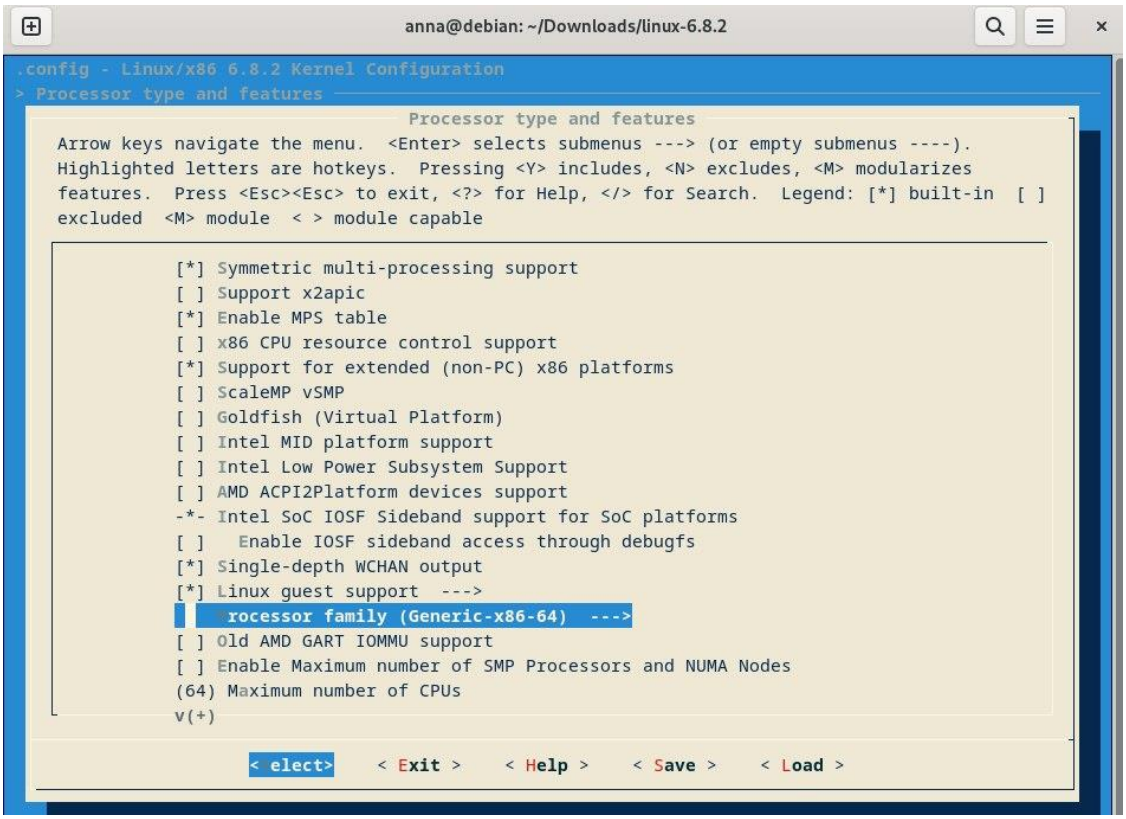
2. Конфигурация ядра, настройка включаемых компонентов:

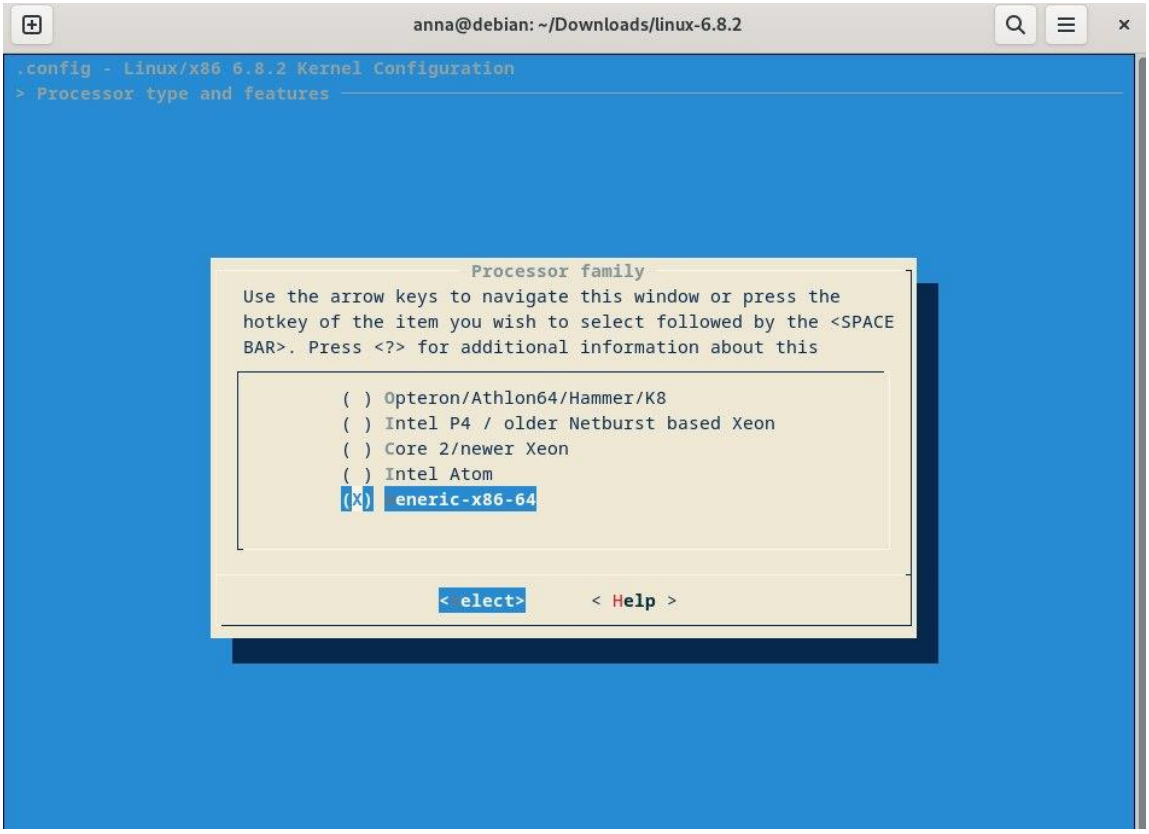
Теперь необходимо настроить сборку под конкретную микроархитектуру. В моем случае это amd64.

make menuconfig

Выбираем свой процессор. В моем случае будет следующая последовательность действий:


anna@debian: ~/Downloads/linux-6.8.2
.config - Linux/x86 6.8.2 Kernel Configuration
Linux/x86 6.8.2 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
excluded <M> module < > module capable
General setup --->
[*] 64-bit kernel
Processor type and features --->
[*] Mitigations for speculative execution vulnerabilities --->
Power management and ACPI options --->
Bus options (PCI etc.) --->
Binary Emulations --->
[*] Virtualization --->
General architecture-dependent options --->
[*] Enable loadable module support --->
-*- Enable the block layer --->
Executable file formats --->
Memory Management options --->
[*] Networking support --->
Device Drivers --->
File systems --->
Security options --->
-*- Cryptographic API --->
v(+)
< elect>
< Exit >
< Help >
< Save >
< Load >


anna@debian: ~/Downloads/linux-6.8.2
.config - Linux/x86 6.8.2 Kernel Configuration
> Processor type and features
Processor type and features
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]
excluded <M> module < > module capable
[*] Symmetric multi-processing support
[ ] Support x2apic
[*] Enable MPS table
[ ] x86 CPU resource control support
[*] Support for extended (non-PC) x86 platforms
[ ] ScaleMP vSMP
[ ] Goldfish (Virtual Platform)
[ ] Intel MID platform support
[ ] Intel Low Power Subsystem Support
[ ] AMD ACPI2Platform devices support
-*- Intel SoC IOSF Sideband support for SoC platforms
[ ] Enable IOSF sideband access through debugfs
[*] Single-depth WCHAN output
[*] Linux guest support --->
Processor family (Generic-x86-64) --->
[ ] Old AMD GART IOMMU support
[ ] Enable Maximum number of SMP Processors and NUMA Nodes
(64) Maximum number of CPUs
v(+)
< elect>
< Exit >
< Help >
< Save >
< Load >



3. Сборка ядра

В лабораторной работе №6 было установлено, что оптимальное время достигается при сборке ядра на 8 потоках. Соберем полученное нами ядро на 8 потоках, а затем сравним полученные результаты времени сборки с данными, полученными в лабораторной работе №6:

make -j8

Завершающим этапом является установка сконфигурированного ядра, представленная в лабораторной работе №6.

4. Сравним полученные результаты:

Сборка ядра (лаб. №6) – 441 сек.

Сборка ядра под целевую платформу – 396 сек.

Мы получили, что при отключении графической информации и сборке ядра под целевую платформу время сборки ядра сократилось.

3. Завершение

3.1 Выводы по результатам проделанной работы:

1. Поставленная цель была достигнута; Ядро, собранное под данную архитектуру успешно установилось и запустилось.

2. Поставленные задачи были решены

3. Сделанные выводы:

Сборка ядра Linux под целевую платформу позволяет настроить ядро под конкретную систему, что обеспечивает эффективную работу устройства.