МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«МИРЭА – РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»
РТУ МИРЭА

Институт ИКБ

Специальность (направление): 09.03.02 (информационные системы и технологии)

Кафедра: КБ-3 «Разработка программных решений и системного программирования»

Дисциплина: «Безопасность операционных систем»

Практическая работа
на тему:
Обработка прерываний

Студент: _____ 15.11.2024 Крашенинников М.В.
подпись *Дата* *инициалы и фамилия*

Группа: БСБО-16-23 Шифр: 23Б0107

Преподаватель: _____ 15.11.2024 Иванова И.А.
подпись *дата* *инициалы и фамилия*

Москва 2024 г.

Задание 1:

```python
def main():
    print("Введите что-нибудь, чтобы прервать выполнение.")
    while True:
        if input():
            print("клавиша была нажата! Программа завершает работу.")
            break

if __name__ == "__main__":
    main()
```

```
twox@Matvey:/mnt/c/Users/boltf/OneDrive/Рабочий стол/Безопасность ОС/практика 6$ python3 1.py
Введите что-нибудь, чтобы прервать выполнение.
132
клавиша была нажата! Программа завершает работу.
```

Задание 2:

```python
import time
import queue
import threading

class KeyboardBuffer:
    def __init__(self, frequency):
        self.buffer = queue.Queue()
        self.frequency = frequency
        self.stop_event = threading.Event()

    def add_signal(self, signal):
        """Метод для добавления сигнала в буфер."""
        self.buffer.put(signal)

    def transmit_signals(self):
        """Метод для передачи сигналов с заданной частотой."""
        while not self.stop_event.is_set():
```

```
            if not self.buffer.empty():
                signal = self.buffer.get()
                print(f"Передан сигнал: {signal}")
            time.sleep(1 / self.frequency)

    def stop_transmitting(self):
        """Метод для остановки передачи сигналов."""
        self.stop_event.set()

def simulate_keyboard_input(keyboard_buffer):
    keys = ['a', 'b', 'c', 'd', 'e']
    for key in keys:
        keyboard_buffer.add_signal(key)
        time.sleep(2)

if __name__ == "__main__":
    frequency = 2
    keyboard_buffer = KeyboardBuffer(frequency)

    transmission_thread = threading.Thread(target=keyboard_buffer.transmit_signals)
    transmission_thread.start()

    simulate_keyboard_input(keyboard_buffer)
    time.sleep(3)

    keyboard_buffer.stop_transmitting()
    transmission_thread.join()
    print("Передача завершена.")
```

```
twox@Matvey:/mnt/c/Users/boltf/OneDrive/Рабочий стол/Безопасность ОС$ "/mnt/c/Users/boltf/OneDrive/Рабочий стол/Безопасность ОС/.venv/bin/python" "/mnt/c/Users/boltf/OneDrive/Рабочий стол/Безопасность ОС/практика 6/2.py"
Передан сигнал: a
Передан сигнал: b
Передан сигнал: c
Передан сигнал: d
Передан сигнал: e
Передача завершена.
```

Задание 3:

```
import curses

def emergency_action(stdscr):
    """Функция для выполнения аварийных действий."""
    stdscr.clear()
    stdscr.addstr(0, 0, "Аварийные действия: система остановлена!")
    stdscr.refresh()
    curses.napms(2000)

def keyboard_handler(stdscr):
    """Функция для запуска обработки нажатий клавиш с использованием curses."""
    curses.curs_set(0)
    stdscr.clear()
    stdscr.addstr(0, 0, "Программа работает. Нажмите F10 для аварийного
завершения или ESC для выхода.")
    stdscr.refresh()

    while True:
        key = stdscr.getch()

        if key == 27:
            stdscr.clear()
            stdscr.addstr(1, 0, "Выход по ESC.")
            stdscr.refresh()
            curses.napms(2000)
            break

        if key == curses.KEY_F10:
            stdscr.clear()
            stdscr.addstr(1, 0, "Нажата горячая клавиша F10.")
            stdscr.refresh()
```

```
            emergency_action(stdscr)
            break

def main():
    """Главная функция для запуска программы с curses."""
    curses.wrapper(keyboard_handler)

if __name__ == "__main__":
    main()
```

```
Программа работает. Нажмите F10 для аварийного завершения или ESC для выхода.
```

```
Выход по ESC.
```

Задание 4:

```
import signal
import time
from datetime import datetime

x = 20
total_time = 0

def handler(signum, frame):
    global x, total_time
    current_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    total_time += x
    print(f"Прошло {x} секунд! Текущее время: {current_time}")
```

```
    x += 20

signal.signal(signal.SIGALRM, handler)
signal.setitimer(signal.ITIMER_REAL, x, 20)

try:
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    print(f"Программа завершена")
```

```
(.venv) twox@Matvey:/mnt/c/Users/boltf/OneDrive/Рабочий стол/Безопасность ОС$ "/mnt/c/Users/boltf/OneDri
ve/Рабочий стол/Безопасность ОС/.venv/bin/python" "/mnt/c/Users/boltf/OneDrive/Рабочий стол/Безопасность
 ОС/практика 6/4.py"
Прошло 20 секунд! Текущее время: 2024-12-17 17:58:51
Прошло 40 секунд! Текущее время: 2024-12-17 17:59:09
^CПрограмма завершена
```

Вопросы:
1. Принципы организации систем прерывания программ.

Прерывание — это сигнал процессору, который заставляет временно приостановить выполнение текущей программы и перейти к выполнению специальной подпрограммы, называемой обработчиком прерывания. Оно нужно для того, чтобы система могла оперативно реагировать на события, такие как завершение ввода-вывода, возникновение ошибок или срабатывание таймера, без постоянной проверки их состояния.

2. Вектор прерываний.

Прерывания делятся на два основных вида:

- Аппаратные прерывания:
  - Генерируются устройствами (например, клавиатурой, таймером).
  - Они асинхронны и происходят независимо от выполнения программы.

Программные прерывания:

- Генерируются инструкциями программы.
- Используются для вызова системных функций или переключения между задачами.

3. Основные характеристики систем прерывания.обработать.

3. Чем отличаются аппаратные и программные прерывания?

Характеристика	Аппаратные прерывания	Программные прерывания
Источник	Физические устройства (например, таймер).	Инструкции внутри программы.
Асинхронность	Да (могут произойти в любое время).	Нет (вызываются программой).
Пример	Сигнал от клавиатуры, завершение ввода-вывода.	Вызов системного вызова через таблицу.

4.Типы приоритетов прерываний.

1. Жёсткий (фиксированный) приоритет:

   Приоритеты прерываний определены на этапе проектирования системы и не меняются во время работы.

   Например, прерывание от таймера может всегда иметь приоритет выше, чем от клавиатуры.

2. Динамический (изменяемый) приоритет:

   Приоритеты прерываний могут изменяться во время выполнения программы.

   Используется для адаптации к текущим условиям работы системы.

3. Каскадный приоритет:

   Если происходит несколько прерываний одновременно, система обрабатывает их в порядке предопределённого списка.

4.Прерывания без приоритета:

   Все прерывания обрабатываются по мере поступления. Если два прерывания происходят одновременно, обрабатывается то, что поступило первым.

5. Вложенные прерывания.

Вложенные прерывания — это ситуация, когда во время обработки одного прерывания поступает другое, более приоритетное. В таком случае:

1) Текущее состояние обработчика первого прерывания сохраняется (например, в стеке).
2) Система переключается на выполнение обработчика нового (более приоритетного) прерывания.
3) После завершения обработки второго прерывания выполнение первого продолжается с места, где оно было прервано.

6. Типы прерываний.

1) Аппаратные прерывания:

   Генерируются внешними устройствами.

Пример: сигнал от клавиатуры, завершение передачи данных через сеть.

2) Программные прерывания:

Генерируются инструкциями программы.

Пример: вызов системных функций через таблицу векторов прерываний.

3) Синхронные прерывания:

Возникают в строго определённые моменты выполнения программы.

Пример: деление на ноль, ошибка доступа к памяти.

4) Асинхронные прерывания:

Происходят независимо от выполнения программы.

Пример: сигнал от таймера или внешнего устройства.

5) Маскируемые и немаскируемые прерывания:

Маскируемые — можно временно отключить (например, прерывания от клавиатуры).

Немаскируемые — всегда обрабатываются (например, сигнал о сбое питания).

6) Внешние и внутренние прерывания:

Внешние — вызваны внешними устройствами.

Внутренние — вызваны самим процессором (например, ошибки исполнения).