Министерство цифрового развития, связи и
массовых коммуникаций Российской Федерации
СибГУТИ

Отчёт по информатике
«Исследование иерархии памяти, работы процессора в ЭВМ на языке Си»

Выполнил студент: Родионов П.А.
группы: ИВ-522
Вариант 9
Проверил ассистент кафедры ВС: Шевченко М.В.

Новосибирск 2025 г.

Теоретические ответы + команды + вывод терминала

Буферизация ввода и её влияние

В Unix-терминале по умолчанию работает *канонический режим*.
Это означает:

- символы не передаются программе сразу;
- ввод буферизуется в терминале до нажатия Enter;
- `getchar()` получает данные только после завершения строки.

Поэтому в программе использование:

```
setbuf(stdin, NULL)
```

отключает лишь *внутренний буфер stdio*, но **не отключает канонический режим терминала**, поэтому `getchar()` продолжает ждать Enter.

Чтобы увидеть разницу, я запускал программу с разными настройками TTY:

```
stty -icanon
```

— отключает канонический режим

```
stty -icanon -echo
```

— отключает ввод построчно + отображение набранных символов

И после этого вывод программы изменился — теперь ввод передаётся сразу, без Enter.

Листинг программы — что в ней происходит

Где происходит нагрузка на CPU и RAM

Нагрузка создаётся в функции:

```
unsigned long func(int n)
```

Внутри неё:

- выполняется цикл до `n` (до 100 000), что даёт нагрузки на CPU;
- создаются два больших массива:
    - `mem[65536]` — 64 КБ, нагружает L1/L2;
    - `data[256 * 1024]` — 256 КБ, выходит за пределы L2, попадает в L3 и RAM.

В цикле постоянно меняются:

```
mem[i % 65536]
data[i % (256 * 1024)]
```

Это заставляет процессор:

- обращаться к кеш-линиям L1/L2/L3;
- периодически выгружать данные в RAM;
- выполнять тысячи операций записи.

То есть функция **специально имитирует смешанную нагрузку процессор + память**, чтобы показать влияние иерархии памяти.

Как реализована «защита»

Защита — это механизм **контроля ввода**:

- проверка, что введены только цифры;
- ограничение диапазона 1–100 000;
- повторный запрос при ошибке;
- защита от неверного ввода (буквы, символы).

Также скрытая защита:

- если EOF → программа завершится корректно;
- буферизация может быть отключена, что нужно для проверки поведения программы.

Как учтена иерархия памяти

Иерархия памяти проявляется в функции `func()`:

- массив 64 КБ помещается в **L1/L2** → быстрые обращения.
- массив 256 КБ помещается в **L3** → медленнее, требуется больше тактов.
- при большом `n` часть доступа уходит в **RAM** → ещё медленнее.

Таким образом:

- первые итерации цикла работают быстро (данные в L1/L2),
- средние — чуть медленнее (L3),
- последние — самые медленные (RAM).

Это позволяет наблюдать, что чем больше массив и объём данных, тем медленнее вычисление.

Как работает шифрование (уровень 5)

Функция шифрования:

```
void encr(unsigned long sum, char *cpum)
```

Использует **первый символ имени процессора как ключ**:

```
key = cpum[0]
```

Каждая цифра результата шифруется:

```
(digit + key) % 10
```

То есть:

- для разных процессоров → разные ключи;
- результат нельзя получить без исходного `cpum[0]`.

Это простая симметричная схема, но она **привязана к конкретной машине**, что требуется по условию.

Пример выполнения

```
[oniic@archlinux procmemorylearning]$ ./run

Enter proc name: AMUDE
Buff: 0
buff is on
Enter num: 1-100 000
Num: 12

result: 78
enc result: 25
[oniic@archlinux procmemorylearning]$ ./run

Enter proc name: AMUDE
Buff: 1
buff is off

Enter num: 1-100 000
Num: 12

result: 78
enc result: 25
[oniic@archlinux procmemorylearning]$ stty -icanon -echo
[oniic@archlinux procmemorylearning]$

Enter proc name: Buff: buff is off
Enter num: 1-100 000
Num:
result: 78
enc result: 25
[oniic@archlinux procmemorylearning]$
[oniic@archlinux procmemorylearning]$ stty -icanon echo
[oniic@archlinux procmemorylearning]$ ./run

Enter proc name: AMUDE
Buff: 1buff is off

Enter num: 1-100 000
Num: 12

result: 78
enc result: 25
[oniic@archlinux procmemorylearning]$
```

Вывод

Почему программа работает только на моей машине?
Шифрование использует первый символ строки с названием процессора (`cpum[0]`), поэтому зашифрованный результат зависит от конкретной машины.
Кроме того, скорость выполнения цикла зависит от частоты процессора, объёма и скорости кэшей, поэтому на разных ПК программа работает по-разному.

Как иерархия памяти влияет на производительность?
При обращении к данным в массиве 64 КБ используется кэш L1/L2 → доступ очень быстрый.
Массив 256 КБ частично попадает в L3 → доступ медленнее.
Когда данные не помещаются в кэш, процессор обращается к RAM, что увеличивает задержки в десятки раз.
Поэтому время выполнения возрастает при больших `n`.

Что я узнал о буферизации ввода?
Я увидел, что:

- `setbuf(stdin, NULL)` не выключает канонический режим терминала;
- настоящий контроль ввода делается через настройки TTY (`stty -icanon`);
- в каноническом режиме терминал сам буферизует строку и передаёт её только после Enter;
- после отключения канонического режима ввод происходит посимвольно.