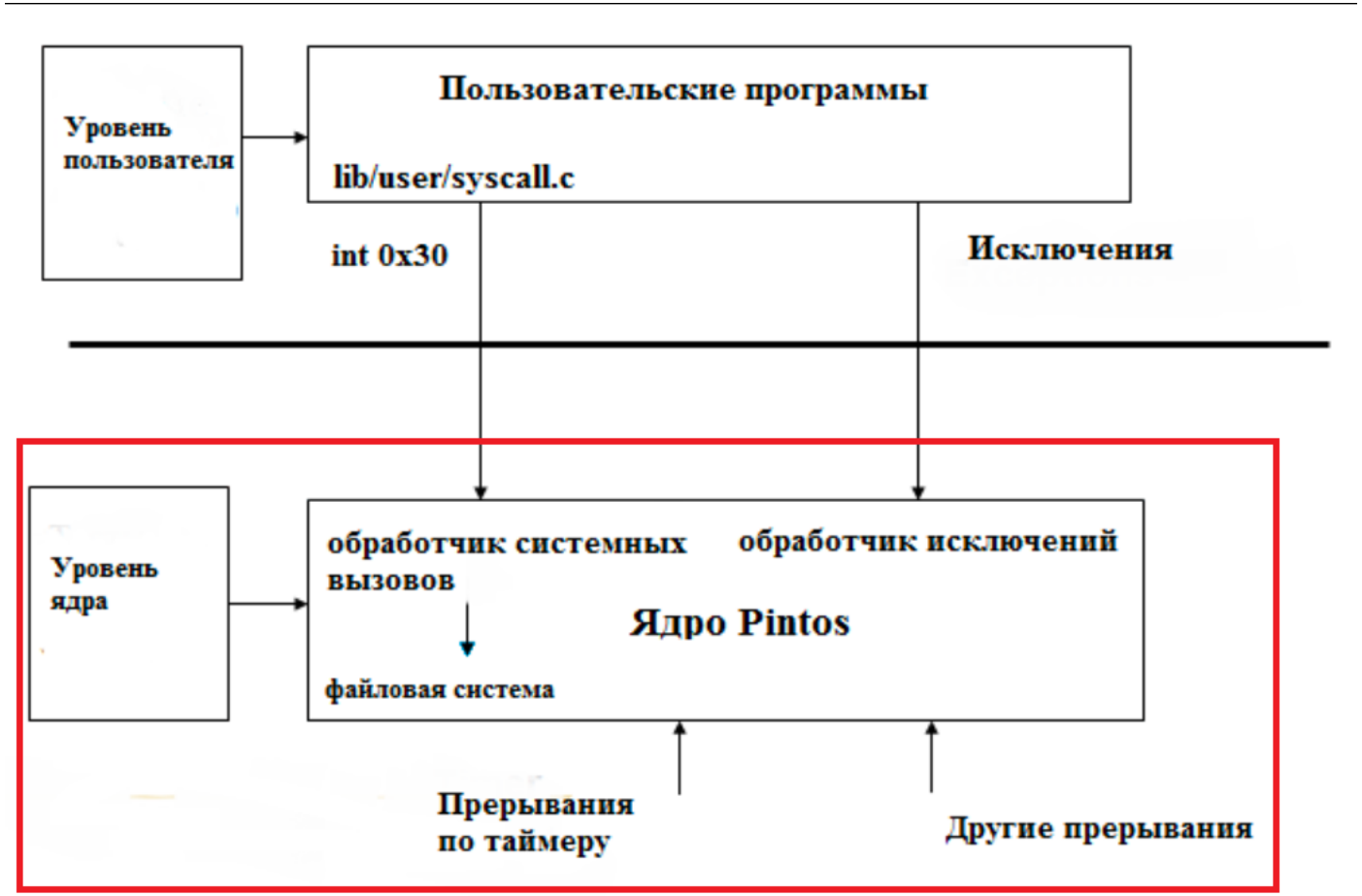
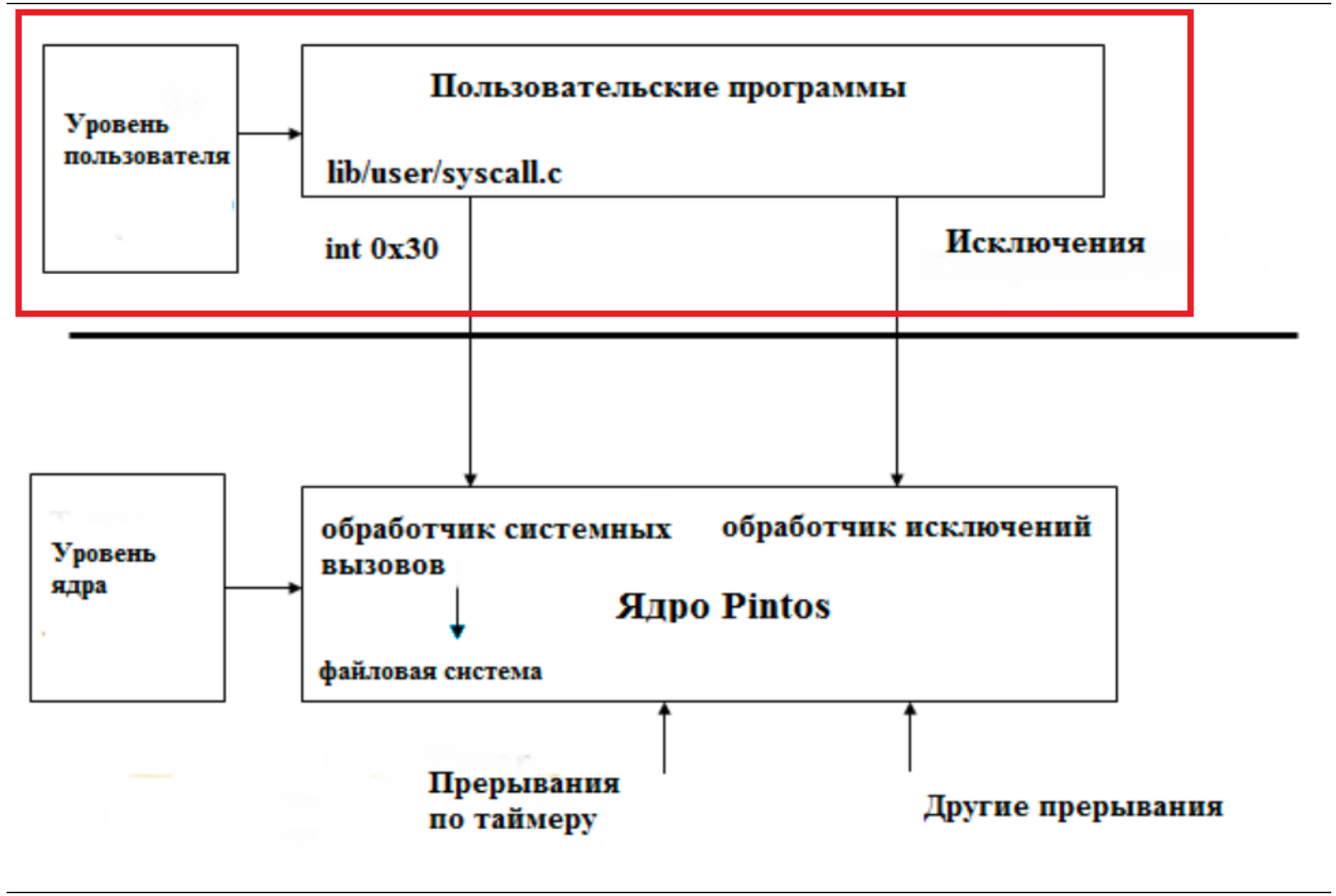
Четвертая работа

Уровни взаимодействия с ОС Pintos



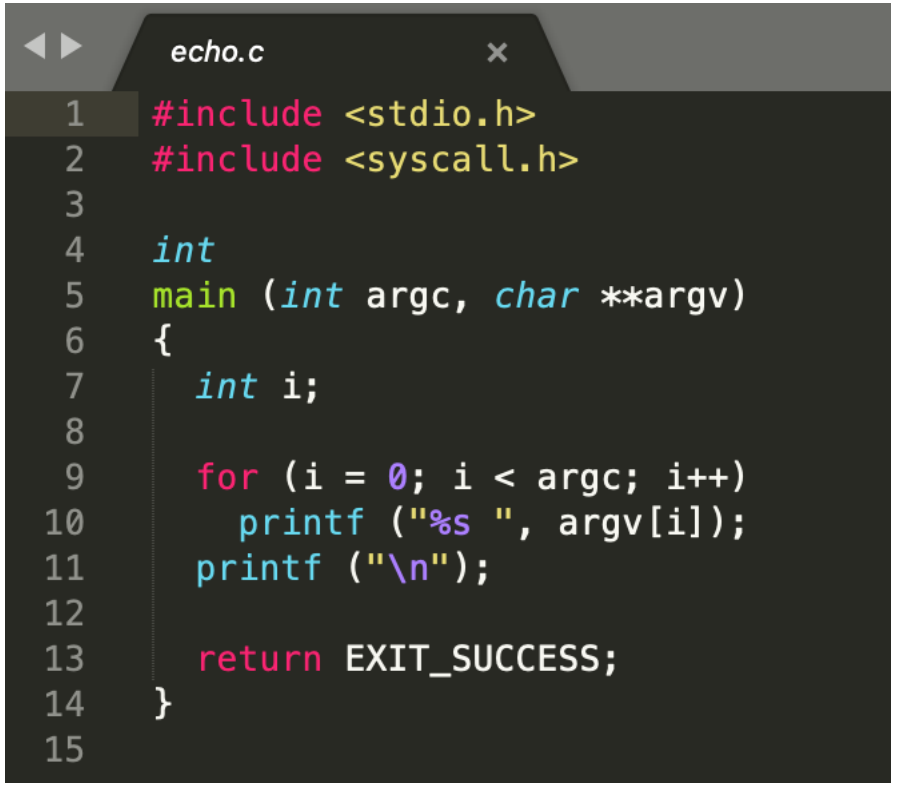
Работали на этом уровне на протяжении 1-3 лабораторных работ

Уровни взаимодействия с ОС Pintos



Необходимо внедрить в систему возможность обработки пользовательских программ

Пользовательские программы



ОС Pintos может выполнять программы, разработанные на языке программирования C, если:

- они не превышают заданный лимит по размеру
- используют ограниченный набор системных вызовов (например, не используют динамическое выделение памяти)
- не содержат вычислений над числами с плавающей запятой

Файловая система Pintos

Пользовательские программы **не могут быть загружены из хостовой машины**, а должны быть скопированы на виртуальный диск с файловой системой

Примечание: файловая система ОС Pintos имеет очень ограниченный функционал

Пример запуска пользовательской программы *echo* в ОС Pintos:

pintos --qemu --disk=filesys.dsk -p ../../examples/echo -a echo -- -q — запись программы на виртуальный диск
pintos --qemu --disk=filesys.dsk -- run 'echo x' — запуск программы с виртуального диска

```
Boot complete.
Executing 'echo x':
Execution of 'echo x' complete.
Timer: 73 ticks
Thread: 0 idle ticks, 73 kernel ticks, 0 user ticks
```

Программа не была исполнена корректно, т.к. в системе отсутствуют:

1. механизм ожидания завершения запущенной программы
2. механизм передачи параметров командной строки запускаемой программы

Механизм ожидания завершения запущенной программы

Необходимо изучить детали реализации загрузки пользовательских программ в память:

- какие функции выполняет процесс-родитель
- какие функции выполняет процесс-потомок
- как они взаимодействуют процесс-родитель и процесс-потомок
- как выглядит жизненный цикл процесса

```
/* Waits for thread TID to die and returns its exit status.  If
   it was terminated by the kernel (i.e. killed due to an
   exception), returns -1.  If TID is invalid or if it was not a
   child of the calling process, or if process_wait() has already
   been successfully called for the given TID, returns -1
   immediately, without waiting.

   This function will be implemented.  For now, it
   does nothing. */
int
process_wait (tid_t child_tid UNUSED)
{
  return -1;
}
```

Функция *process_wait* необходима для того, чтобы **ядро дождалось завершения всех процессов**

В ходе работы требуется реализовать эту логику

Механизм передачи параметров командной строки запускаемой программы

```
bool load (const char *file_name, void (**eip) (void), void **esp)
{
  int success = false;

  /* Open executable file. */
  file = filesys_open (file_name);
  if (file == NULL)
    {
      printf ("load: %s: open failed\n", file_name);
      goto done;
    }

  /* Проверка корректности исполняемого файла */

  /* Read program headers. */
  for (i = 0; i < ehdr.e_phnum; i++)
    {
      file_seek (file, file_ofs);

      if (file_read (file, &phdr, sizeof phdr) != sizeof phdr)
        goto done;

      /* Загрузка исполняемого файла с диска */
    }

  /* Set up stack. */
  if (!setup_stack (esp))
    goto done;

  /* ??? */

  success = true;

 done:
  /* We arrive here whether the load is successful or not. */
  file_close (file);
  return success;
}
```

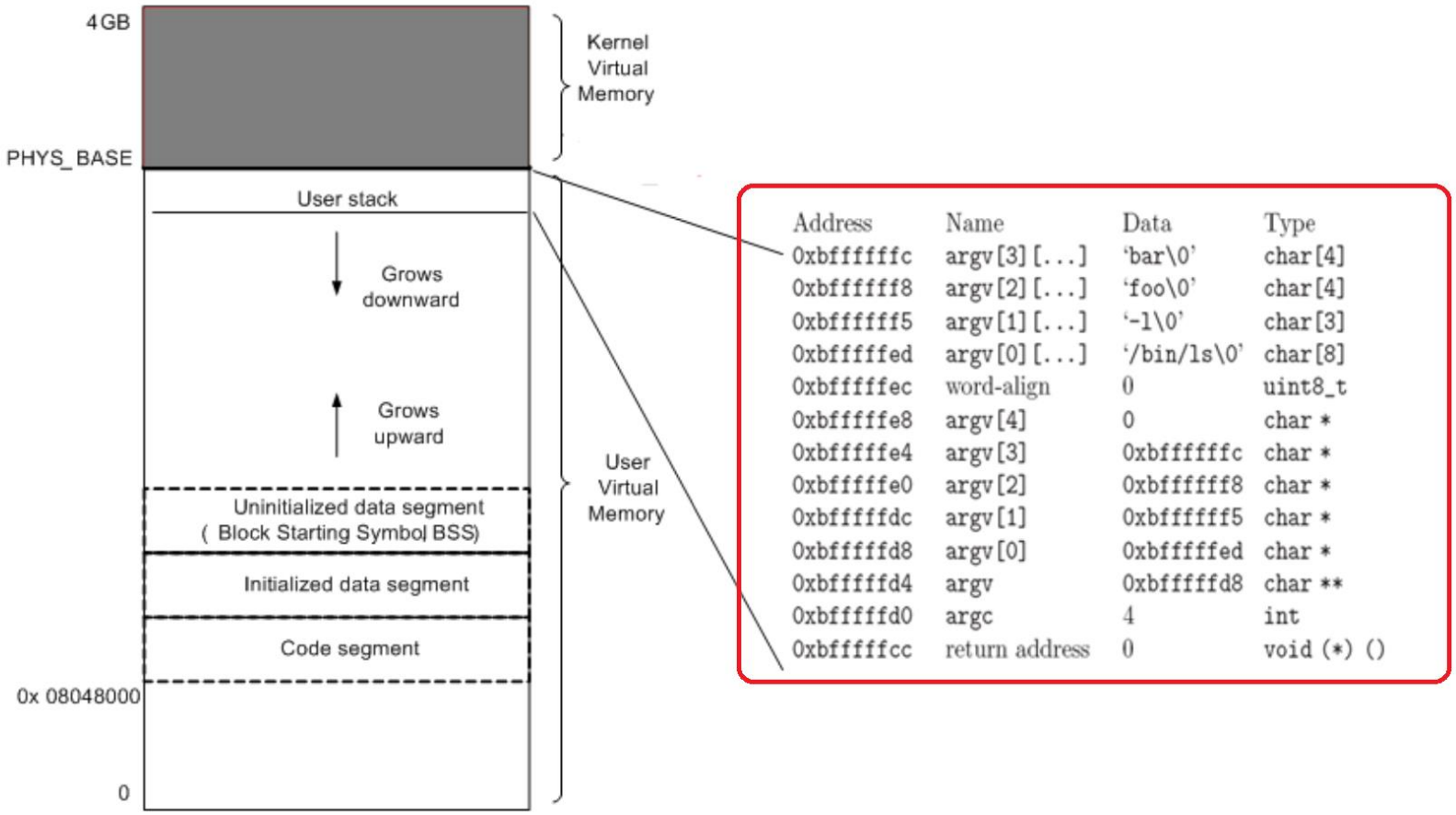
Функция *load* выполняет загрузку исполняемого файла с диска в оперативную память

В базовой реализации не настроена передача аргументов командной строки

Необходимо:

- изучить как выполняется обработка параметров командной строки (например, *echo x*)
- изучить устройство виртуальной памяти в ОС Pintos
- поместить аргументы в стек в правильном порядке

Виртуальное адресное пространство



Address	Name	Data	Type
0xbffffffc	argv[3][...]	'bar\0'	char[4]
0xbffffff8	argv[2][...]	'foo\0'	char[4]
0xbffffff5	argv[1][...]	'-l\0'	char[3]
0xbfffffed	argv[0][...]	'/bin/ls\0'	char[8]
0xbfffffec	word-align	0	uint8_t
0xbfffffe8	argv[4]	0	char *
0xbfffffe4	argv[3]	0xbffffffc	char *
0xbfffffe0	argv[2]	0xbffffff8	char *
0xbfffffdc	argv[1]	0xbffffff5	char *
0xbfffffd8	argv[0]	0xbfffffed	char *
0xbfffffd4	argv	0xbfffffd8	char **
0xbfffffd0	argc	4	int
0xbfffffcc	return address	0	void (*) ()

Примечание: стек заполняется в обратном порядке, от наибольших адресов памяти в сторону уменьшения адресов, «снизу вверх»

Задачи

1. Модифицировать работу системы обработки команд пользователя и передачи параметров командной строки так, чтобы пользовательская программа могла считывать переданные ей аргументы

2. Реализовать функцию ожидания завершения пользовательских программ

3. Реализовать минимальную поддержку системных вызов (write, exit) от пользовательских программ к ядру ОС pintos