Практическое задание № 3. Консоль управления моделью Simple Computer. Текстовая часть.

Цель работы

Изучить принципы работы терминалов ЭВМ в текстовом режиме. Понять, каким образом кодируется текстовая информация и как с помощью неё можно управлять работой терминалов. Разработать библиотеку функций myTerm, включающую базовые функции по управлению текстовым терминалом. Доработать устройство ввода-вывода Simple Computer (вывести на экран текстовую часть консоли).

Задание на лабораторную работу

1. Прочитайте главу 5 практикума по курсу «Организация ЭВМ и систем» (читать тут). Обратите особое внимание на параграфы 5.4 и 5.5. Изучите страницу `man` для команды `infocmp`, базы `terminfo`, функции `ioctl`.
2. Откройте текстовый терминал и запустите оболочку `bash` (оболочка запускается автоматически). Используя команду `infocmp`, определите (и перепишите их себе) escape-последовательности для терминала, выполняющие следующие действия:
   - очистка экрана и перемещение курсора в левый верхний угол (clear_screen);
   - перемещение курсора в заданную позицию экрана (cursor_address);
   - задание цвета последующих выводимых символов (set_a_background);
   - задание цвета фона последующих выводимых символов (set_a_foreground);
   - скрытие и восстановление курсора (cursor_invisible, cursor_visible);
   - очистка текущей строки (delete_line).
3. Используя оболочку bash, команду echo –e и скрипт[1], проверьте работу полученных последовательностей. Символ escape задается как \033 или \E. Например – echo -e "\033[m". Для проверки сформируйте последовательность escape-команд, выполняющую следующие действия:
   - очищает экран;
   - выводит в пятой строке, начиная с 10 символа Ваше имя красными буквами на черном фоне;
   - в шестой строке, начиная с 8 символа Вашу группу зеленым цветом на белом фоне;
   - перемещает курсор в 10 строку, 1 символ и возвращает настройки цвета в значения «по умолчанию».

   Скрипт должен располагаться в корневом каталоге проекта и называться test_terminal.sh.
4. Разработать функции библиотеки myTerm:
   - `int mt_clrscr (void)` - производит очистку и перемещение курсора в левый верхний угол экрана;
   - `int mt_gotoXY (int, int)` - перемещает курсор в указанную позицию. Первый параметр номер строки, второй - номер столбца;
   - `int mt_getscreensize (int * rows, int * cols)` - определяет размер экрана терминала (количество строк и столбцов). Если определение невозможно, то статус возврата = -1, иначе 0;
   - `int mt_setfgcolor (enum colors)` - устанавливает цвет последующих выводимых символов. В качестве параметра передаётся константа из созданного Вами перечислимого типа `colors`, описывающего цвета терминала;

[1] Скрипт – это текстовый файл, содержащий команды оболочки. Запускается на выполнение командой bash имя_файла.

- `int mt_setbgcolor (enum colors)` - устанавливает цвет фона последующих выводимых символов. В качестве параметра передаётся константа из созданного Вами перечислимого типа `colors`, описывающего цвета терминала.
- `int mt_setdefaultcolor (void)` – устанавливает цвета символов и фона в значения по умолчанию;
- `int mt_setcursorvisible (int value)` – скрывает или показывает курсор;
- `int mt_delline (void)` – очищает текущую строку

Все функции возвращают 0 в случае успешного выполнения и -1 в случае ошибки. В качестве терминала используется стандартный поток вывода. Вывод управляющих последовательностей должен быть реализован с использованием низкоуровневого вывода (функция write).

5. Оформите разработанные функции как статическую библиотеку myTerm. Подготовьте заголовочный файл для неё. Доработайте систему сборки приложения таким образом, чтобы статическая библиотека myTerm собиралась при изменении любого из файлов с исходным кодом. Собранная библиотека должна располагаться в каталоге myTerm.
6. Доработайте устройство ввода-вывода:
   - Измените функцию `printCell` -> `void printCell (int address, enum colors fg, enum colors bg)`. Теперь эта функция выводит значение ячейки памяти с учетом заданных цветов символов и фона. Кроме этого местоположение выводимого значения рассчитывается исходя из заданного адреса ячейки памяти и расположения блока «Оперативная память»;
   - Измените функцию `void printFlags (void)`. Теперь она выводит значения флагов в нужном месте экрана (блок «Регистр флагов»);
   - Измените функцию `void printDecodedCommand (int value)`. Теперь она выводит значения в нужно месте экрана (блок «Редактируемая ячейка (формат)»).
   - Измените функцию `void printAccumulator (void)`. Теперь она выводит значение в нужном месте экрана (блок «Аккумулятор»), значения выводятся в декодированном виде и в шестнадцатеричной системе счисления;
   - Измените функцию `void printCounters (void)`. Теперь она выводит значение счетчика команд в нужном месте экрана (блок «Счетчик команд») и в соответствующем формате;
   - Разработайте функцию `void printTerm (int address, int input)` - которая выводит очередную строку в блок “IN—OUT”, address – это адрес выводимой ячейки памяти, а input – это признак ввода значения или его вывода. Если значение должно быть выведено, то оно выводится. Если значение необходимо ввести, то выводится только адрес ячейки и признак ввода (>). Функция должна реализовать механизм «прокрутки» строк, т.к. выводить текущее значение и максимум три предыдущих;
   - Разработайте функцию `void printCommand (void)` - которая выводит результат декодирования ячейки памяти, адрес которой указан в счетчике команд. Если команда неверная, то перед результатом декодирования должен выводиться знак «!». Значение должно выводиться в соответствующем месте экрана (блок «Команда»);
7. Разработайте приложение console, которое:
   - При запуске приложения проверялось, что поток вывода соответствует терминалу. Если это не так, что приложения принудительно завершается;

- Проверяется размер экрана терминала. Если размера не хватает для того, чтобы вывести консоль (с учетом псевдографических символов), то на экран выводится сообщение об ошибке и программа завершается;
- Экран терминала очищается;
- На экран выводятся все текстовые данные консоли (кроме рамок, блока с увеличенным значением текущей редактируемой ячейки памяти, заголовков блоков). Одна ячейка памяти в блоке «Оперативная память» должна быть выведена в инверсном режиме (это «текущая редактируемая ячейка»).
- В блок “IN—OUT” последовательно выводятся 7 значений произвольных ячеек оперативной памяти.

8. Доработайте Makefile таким образом, чтобы автоматизированная сборка дополнительно смогла собрать библиотеку myTerm и исполняемый файл concole. Исполняемый файл должен собираться при изменении библиотек (любой из используемых) или любого из исходных файлов каталога console.

Контрольные вопросы

1. Взаимодействие с устройствами в Linux. Специальные файлы устройств.
2. Функции open, close, read, write.
3. Терминалы. Типы терминалов. Эмуляция терминала. Режимы работы.
4. Управление терминалом. Команды. Низкоуровневое управление.
5. Что такое escape-последовательность?
6. Как определить escape-последовательности для терминала?