

Петрозаводский государственный университет
Кафедра информационно-измерительных систем и физической электроники

**ПРОГРАММА ПОИСКА ЧАСТО ИСПОЛЬЗУЕМЫХ КОМАНД USMD
(Инд. задание № 1. Вариант 11)**

Выполнил:
студент 2 курса
Иванов И. И.

Руководитель:
ст. преподаватель
Петров П. П.

Петрозаводск, 2008

СОДЕРЖАНИЕ

1 Техническое задание	3
1.1 Назначение программы	3
1.2 Требования к входным и выходным данным	3
1.3 Требования к информационной совместимости	3
1.4 Требования к техническим средствам	3
1.5 Требования к программной документации	3
1.6 Срок сдачи программы	3
2 Текст программы	4
3 Описание программы	7
3.1 Общее описание	7
3.2 Блок-схема программы	7
3.3 Логическая структура программы	8
3.3.1 Структуры данных	8
3.3.2 Функции	8
3.3.3 Тело программы	8
3.4 Конфигурирование программы	9
3.5 Входные и выходные данные	9
3.5.1 Входные данные	9
3.5.2 Выходные данные	10
3.6 Вызов и загрузка	10
3.6.1 Компиляция	10
3.6.2 Запуск программы	10
3.7 Сообщения программы	10

1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1 Назначение программы

Программа UCMD предназначена для поиска часто используемых команд. Программа анализирует историю команд, формируемую командным интерпретатором BASH, и выводит имена трёх наиболее часто используемых команд.

1.2 Требования к входным и выходным данным

В качестве исходных данных для программы используется файл истории команд `.bash_history`, находящийся в текущем каталоге. Результат работы программы выдаётся в файл стандартного вывода. Диагностические сообщения программы (сообщения об ошибках) выдаются в файл стандартного вывода ошибок.

1.3 Требования к информационной совместимости

Программа разрабатывается на языке Си с использованием программных интерфейсов POSIX и GNU, применяемых в системах на основе ядра Linux. Программа применяется совместно с командным интерпретатором BASH версии 2.0 или более поздней.

1.4 Требования к техническим средствам

Программа должна запускаться на ЭВМ, технические характеристики которой позволяют работать в операционной системе на основе ядра Linux с командным интерпретатором BASH.

1.5 Требования к программной документации

Программная документация должна содержать:

- Текст программы;
- Описание программы.

1.6 Срок сдачи программы

Срок сдачи программы – 14.03.2008.

2 ТЕКСТ ПРОГРАММЫ

```
/*
Программа поиска часто используемых команд UCMD
(C) 2008 Lab127 Team

This program is free software; you can redistribute it and/or modify it under
the terms of the GNU General Public License as published by the Free Software
Foundation; either version 2 of the License, or any later version.
This program is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with
this library; if not, write to the Free Software Foundation, Inc.,
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>

/* Максимальный допустимый размер строки в файле истории: */
#define BUFSIZE 1024

/* Количество команд, которые надо выводить в результате: */
#define MAXUCMD 3

/* Элемент массива использованных команд: */
typedef struct {
    char *cmd;           //имя команды
    int cntr;           //счётчик использований
} history_counter;

/* Функция поиска самой часто используемой команды:
   hc_arr - массив использованных команд
   hc_sz - размер массива использованных команд
   Результат:
   индекс самой часто исп. команды в массиве hc_arr
   При ошибке возвращается -1.
*/
int search_max_hc(history_counter *hc_arr, int hc_sz) {
    int i;
    int idx_max = 0;
    if (!hc_arr || hc_sz<0) return -1;
    for (i=0; i<hc_sz; i++) {
        if (hc_arr[i].cntr > hc_arr[idx_max].cntr) idx_max = i;
    }
    return idx_max;
}

/* Функция поиска команды по имени и увеличения её счётчика:
   hc_arr - массив использованных команд
   hc_sz - размер массива использованных команд
   cmd - имя команды
   Результат:
   0 - если команда не найдена,
   1 - если команда найдена и счётчик изменён
   -1 - при ошибке
*/
int incr_cmd_cntr(history_counter *hc_arr, int hc_sz, char *cmd) {
    int i;
    if (!hc_arr || hc_sz<0 || !cmd) return -1;
    for (i=0; i<hc_sz; i++) {
        if (!strcmp(cmd, hc_arr[i].cmd)) {
            hc_arr[i].cntr++;
            return 1;
        }
    }
    return 0;
}
```

```

/* Функция добавления новой команды в массив исп. команд:
   hc_arr - массив использованных команд
   hc_sz - размер массива использованных команд
   cmd - имя команды
   Результат:
   1 - при успешном добавлении
   -1 - при ошибке
*/
int add_new_cmd(history_counter *hc_arr, int hc_sz, char *cmd) {
    if (!hc_arr || hc_sz<0 || !cmd || strlen(cmd)==0) return -1;
    hc_arr[hc_sz].cmd = strdup(cmd);
    if (!hc_arr[hc_sz].cmd) return -1;
    hc_arr[hc_sz].cntr = 1;
    return 1;
}

/* Функция выделения имени команды в командной строке:
   cmd - командная строка
   Результат:
   1 - имя команды успешно выделено
   -1 - при ошибке
*/
int extract_cmd_name(char *cmd) {
    char *ptr;
    if (!cmd || strlen(cmd)==0) return -1;
    if ((ptr=strchr(cmd, ' ')) *ptr = 0;
    if ((ptr=strchr(cmd, '\n')) *ptr = 0;
    if ((ptr=strchr(cmd, '\t')) *ptr = 0;
    return 1;
}

/* Точка входа в программу: */
int main() {
    int j;
    FILE *f; //дескриптор файла истории
    char buf[BUFSIZE]; //буфер строки файла истории
    int hc_sz = 0; //размер массива исп. команд
    history_counter *hc_arr; //массив использованных команд

    // БЛОК ИНИЦИАЛИЗАЦИИ ПРОГРАММЫ
    hc_arr = malloc(sizeof(history_counter));
    if (!hc_arr) {
        perror("malloc() failed");
        exit(1);
    }

    f = fopen(".bash_history", "r");
    if (!f) {
        perror("fopen() failed");
        exit(1);
    }

    // БЛОК СОЗДАНИЯ МАССИВА ИСПОЛЬЗОВАННЫХ КОМАНД
    while (fgets(buf,BUFSIZE,f)) {
        int found;
        if (extract_cmd_name(buf)<0)
            continue; /* имя не выделено - пустая строка? */
        if (!strlen(buf)) continue;

        found = incr_cmd_cntr(hc_arr, hc_sz, buf);
        if (found == -1) {
            fprintf(stderr, "incr_cmd_cntr() failed()\n");
            exit(1);
        }
        if (found == 1)
            continue; /* команда уже есть в массиве */
    }
}

```

```

/* добавить в массив, если не нашли */
if (add_new_cmd(hc_arr, hc_sz, buf)<0) {
    fprintf(stderr, "add_new_cmd() failed\n");
    exit(1);
}

/* увеличиваем размер массива */
hc_sz++;
hc_arr = realloc(hc_arr, (hc_sz+1)*sizeof(history_counter));
if (!hc_arr) {
    perror("realloc() failed");
    exit(1);
}
}
fclose(f);

// БЛОК ПОИСКА НАИБОЛЕЕ ЧАСТО ИСПОЛЬЗУЕМЫХ КОМАНД ПО МАССИВУ
/* выделяем MAXUCMD (или hc_sz) самых часто используемых команды */
for (j=0; j<(hc_sz<MAXUCMD ? hc_sz : MAXUCMD); j++) {
    int idx_max = search_max_hc(hc_arr, hc_sz);
    if (idx_max<0) {
        fprintf(stderr, "search_max_hc() failed\n");
        exit(1);
    }
    printf("%s %d\n", hc_arr[idx_max].cmd, hc_arr[idx_max].cntr);
    hc_arr[idx_max].cntr = 0;
}

// БЛОК ОСВОБОЖДЕНИЯ РЕСУРСОВ ПРОГРАММЫ
for (j=0; j<hc_sz; j++) {
    free(hc_arr[j].cmd);
}
free(hc_arr);

return 0;
}

```

3 ОПИСАНИЕ ПРОГРАММЫ

3.1 Общее описание

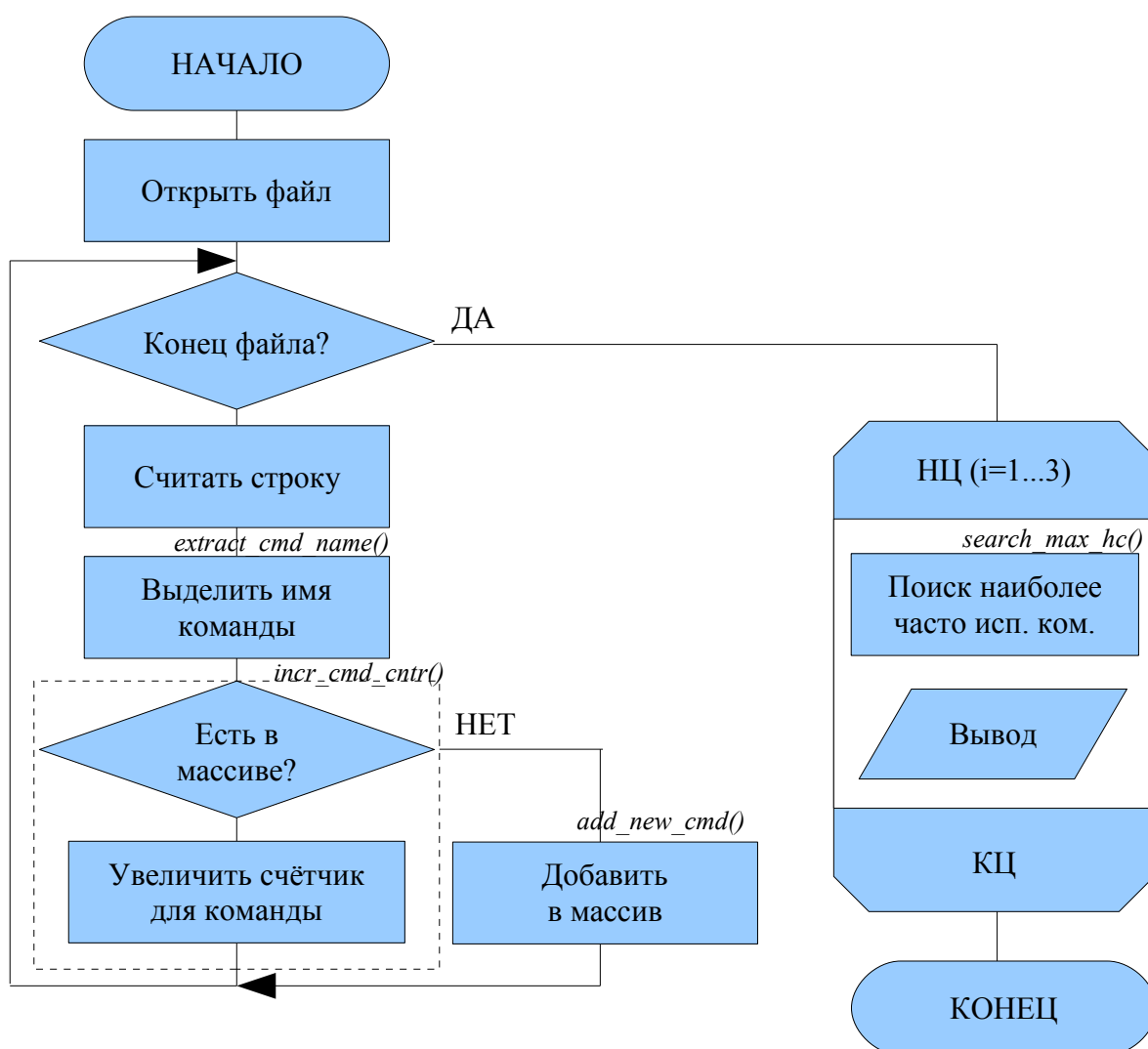
Программа UCMD предназначена для поиска часто используемых команд. Программа анализирует текстовый файл `.bash_history` (историю команд), формируемый командным интерпретатором BASH, и выводит имена трёх наиболее часто используемых команд в порядке уменьшения частоты их использования.

Программа написана на языке Си с использованием программных интерфейсов POSIX и GNU, применяемых в системах на основе ядра Linux. Для компиляции программы использован компилятор GCC версии 4.1. Программа рассчитана на применение совместно с командным интерпретатором BASH версии 2.0 или более поздней.

Программа должна запускаться на ЭВМ, технические характеристики которой позволяют работать в операционной системе на основе ядра Linux с командным интерпретатором BASH.

Правовые условия использования программы – лицензия GPL.

3.2 Блок-схема программы



3.3 Логическая структура программы

3.3.1 Структуры данных

Основная информационная сущность программы – это массив структур `history_counter`. Каждый элемент этого массива представляет собой структуру с двумя полями: именем команды и счётчиком (сколько раз эта команда встретилась в файле истории). Целочисленная переменная, хранящая размер массива, и указатель на первый элемент массива – основные параметры всех функций, работающих с этим массивом.

3.3.2 Функции

3.3.2.1 *search_max_hc()*

Функция `search_max_hc()` ищет в массиве использованных команд `hc_arg` элемент, у которого поле `cntr` имеет максимальное значение.

Функция имеет два входных параметра: `hc_arg` – указатель на первый элемент массива и `hc_sz` – размер массива.

В качестве результата возвращается индекс элемента с максимальным значением `cntr`.

Функция проверяет корректность входных параметров. Если передан недействительный указатель `hc_arg` или отрицательное число `hc_sz`, функция возвращает `-1`, сигнализируя об ошибке.

3.3.2.2 *incr_cmd_cntr()*

Функция `incr_cmd_cntr()` ищет в массиве использованных команд `hc_arg` элемент, соответствующей команде с именем `cmd`. Если такой элемент найден, счётчик `cntr` для этого элемента увеличивается на единицу.

Функция имеет три входных параметра: `hc_arg` – указатель на первый элемент массива, `hc_sz` – размер массива и `cmd` – имя команды.

Если команда с указанным именем найдена в массиве, то функция возвращает `1`, если не найдена – `0`. Функция проверяет также корректность входных параметров. Если передан недействительный указатель `hc_arg` или `cmd`, отрицательное число `hc_sz`, то функция возвращает `-1`, сигнализируя об ошибке.

3.3.2.3 *add_new_cmd()*

Функция добавляет новый элемент в массив использованных команд `hc_arg`. Для работы этой функции необходимо, чтобы `hc_arg` указывал на область памяти, в которой, помимо уже размещённого массива, поместился бы ещё один элемент. Память под копию строки `cmd` функция отводит сама.

Функция имеет три входных параметра: `hc_arg` – указатель на первый элемент массива, `hc_sz` – размер массива (индекс нового элемента) и `cmd` – имя добавляемой команды.

При успешном добавлении функция возвращает `1`. Если же память под копию строки `cmd` не удалось отвести либо входные параметры некорректны (недействительные указатели `hc_arg` или `cmd`, отрицательное значение `hc_sz` или длина строки `cmd` равна `0`), функция возвращает `-1`, сигнализируя об ошибке.

3.3.2.4 *extract_cmd_name()*

Функция выделяет в переданной командной строке имя команды. В переданном строковом буфере сразу после имени команды помещается символ-ограничитель с кодом `0`. Именем команды считается первое слово в командной строке, ограниченное пробелом (символ с кодом ASCII `32`), табуляцией (символ с кодом ASCII `9`), переводом строки (символ с кодом ASCII `10`) или концом строки (символ-ограничитель с кодом `0`).

В качестве параметра функции передаётся адрес буфера строки с командой. Функция изменяет содержимое этого буфера. При успешном выполнении функция возвращает `1`. Если указатель `cmd` недействительный или длина строки `cmd` равна нулю, функция возвращает `-1`, сигнализируя об ошибке.

3.3.3 Тело программы

Точка входа в программу – стандартная для программ на языке Си функция `main()`. В теле программы можно выделить четыре основных блока:

- блок инициализации программы;
- блок создания массива использованных команд;
- блок поиска наиболее часто используемых команд по массиву;
- блок освобождения ресурсов программы.

3.3.3.1 Блок инициализации программы

В блоке инициализации выполняются два действия. Во-первых, резервируется память под первый элемент массива использованных команд `hc_arg`. Во-вторых, открывается поток ввода, связанный с файлом `.bash_history`, в котором командный интерпретатор BASH формирует историю команд.

3.3.3.2 Блок создания массива использованных команд

В данном блоке через открытый поток ввода построчно считывается история команд BASH до тех пор, пока не будет достигнут конец файла. Очередная строка из этого файла помещается в строковый буфер `buf` размером 1024 байта. При помощи функции `extract_cmd_name()` в строке выделяется имя команды. Затем в массиве `hc_arg` производится поиск такой команды и (при наличии) увеличение соответствующего счётчика – при помощи функции `incr_cmd_cntr()`. Если такой элемент в массиве отсутствует, он добавляется при помощи `add_new_cntr()`. Если был добавлен новый элемент, происходит перерезервирование памяти, отведённой под массив, и увеличение счётчика элементов массива.

Когда чтение файла завершено, соответствующий ему входной поток закрывается.

3.3.3.3 Блок поиска наиболее часто используемых команд по массиву

В соответствии с техническим заданием программа должна вывести три наиболее часто используемых команды. Однако количество различных команд в файле истории может быть меньше, т. е. количество элементов в массиве использованных команд может быть 1 или 2. В этом случае программа выведет меньше трёх команд (1 или 2). Именно такое совокупное условие является критерием количества итераций основного цикла в данном блоке.

В этом цикле в массиве использованных команд `hc_arg` при помощи функции `search_max_hc()` ищется элемент с максимальным значением счётчика. Найденный элемент выводится на экран. Счётчик для этого элемента обнуляется, так что он в следующих итерациях не участвует.

3.3.3.4 Блок освобождения ресурсов программы

В данном блоке для каждого элемента массива использованных команд `hc_arg` освобождается память, зарезервированная функцией `add_new_cmd()` для хранения имени команды. После чего освобождается память от самого массива.

3.4 Конфигурация программы

Программа имеет два параметра для конфигурирования:

- `BUFSIZE` – максимально допустимый размер строки в файле истории (без учёта символа перевода строки и символа-ограничителя строки) – по умолчанию 1024;
- `MAXUCMD` – максимальное количество часто используемых команд, выдаваемое программой, – по умолчанию 3.

Параметры конфигурирования указываются в виде макроопределений в исходном тексте программы.

3.5 Входные и выходные данные

3.5.1 Входные данные

В качестве входных данных для программы используется файл `.bash_history`, в котором командный интерпретатор BASH формирует историю команд. Программа ищет данный файл в текущем на момент запуска каталоге. Файл должен иметь текстовую структуру, т. е. состоять из строк текста, ограниченных символами перевода строки (код ASCII 10). Одна строка текста не должна превышать 1023 символа с символом перевода строки. Именем команды считается первое слово в такой строке.

3.5.2 Выходные данные

Результат работы программы формируется на стандартном выводе. Программа выводит до трёх текстовых строк. В каждой строке выводится имя команды и число, которое обозначает, сколько раз команда встретилась в истории команд. Число отделено от имени команды пробелом. Команды выводятся в порядке уменьшения частоты встречаемости.

3.6 Вызов и загрузка

Программа UCMD распространяется в виде исходного текста на языке Си. Перед началом работы с программой её необходимо скомпилировать.

3.6.1 Компиляция

Поскольку программа состоит из единственного программного модуля, использование специального сценария компиляции не представляется целесообразным.

Для компиляции программы следует использовать компилятор GCC. Формат команды компиляции:

```
gcc -Wall -o ucmd ucmd.c
```

где gcc – имя компилятора (в некоторых системах может быть заменено на cc); -Wall – опция компилятора, включающая вывод всех предупреждающих сообщений; -o ucmd – опция компилятора, указывающая имя выходного файла (в данном случае файл ucmd в текущем каталоге); ucmd.c – имя исходного текста программы (при необходимости следует указать путь к файлу).

В случае успешной компиляции GCC не выдаёт никаких сообщений на экран, а в текущем каталоге появляется файл с именем ucmd (или тем, которое указано в опции -o).

3.6.2 Запуск программы

Для запуска программы следует указать в командной строке путь к исполняемому файлу и его имя. Программа не требует дополнительных параметров в командной строке. Если исполняемый файл находится в текущем каталоге, командная строка программы выглядит следующим образом:

```
./ucmd
```

При завершении программы её статус выхода указывает:

- 0 – программа завершилась успешно;
- 1 – программа завершилась неуспешно.

3.7 Сообщения программы

При возникновении нештатной ситуации программа выдаёт диагностическое сообщение на стандартный вывод ошибок и завершается со статусом выхода 1.

Возможные сообщения перечислены в таблице 3.1

Таблица 3.1 – Сообщения программы

Текст сообщения	Комментарий
malloc failed(): Out of memory	Недостаточно памяти для работы программы
fopen failed(): No such file or directory	Файл .bash_history отсутствует в текущем каталоге
fopen failed(): Permission denied	Файл .bash_history не доступен для чтения
incr_cmd_failed()	Внутренняя ошибка программы
add_new_cmd_failed()	Внутренняя ошибка программы или недостаточно памяти для работы
realloc failed(): Out of memory	Недостаточно памяти для работы программы
search_max_hc failed()	Внутренняя ошибка программы