ПРОГРАММИРОВАНИЕ В СРЕДЕ UNIX

Регулярные выражения

Библиотека **regex**

2017 Кафедра «Управления информационными системами и программирования»

Иванов Е.А.

regex

- *Регулярные выражения* являются способом описания текстовых шаблонов для сопоставления.
- Библиотека regex предназначена для работы с регулярными выражениями на Си.
- POSIX определяет два вида регулярных выражений: базовый и расширенный. Программы типа grep, sed и строчный редактор ed используют базовые регулярные выражения. Программы типа egrep и awk используют расширенные регулярные выражения.

regex

Функции входящие в библиотеку regex дают вам возможность использовать в своих программах любой вид

int **regcomp**(regex_t *preg, const char *regex, int cflags);

int **regexec**(const regex_t *preg, const char *string, size_t nmatch, regmatch_t pmatch[], int eflags);

size_t **regerror**(int errcode, const regex_t *preg,

char *errbuf, size_t errbuf_size);

void **regfree**(regex_t *preg);

regex

Чтобы сопоставить регулярное выражение, нужно сначала *откомпилировать* строчную версию регулярного выражения.

Компиляция преобразует регулярное выражение во внутренний формат.

Затем откомпилированная форма *исполняется* для строки для проверки, совпадает ли она с первоначальным регулярным выражением.

regex

int **regcomp**(regex_t *preg, const char *regex, int cflags)

- Компилирует регулярное выражение regex во внутреннее представление, сохраняя его в структуре **regex_t**, на которую указывает **preg**
- **cflags** контролирует процесс компиляции, ее значение равно 0 или побитовому ИЛИ одного или более флагов из табл

REG_EXTENDED	Использовать расширенные регулярные выражения. По умолчанию используются базовые регулярные выражения
REG_ICASE	Сопоставление regexec() игнорирует регистр символов
REG_NEWLINE	Операторы, заменяющие любой символ, не включают символ конца строки
REG_NOSUB	Информация о начале и конце вложенного шаблона не требуется

regex

int **regexec**(const regex_t *preg, const char *string, size_t nmatch, regmatch_t pmatch[], int eflags)

- Выполняет откомпилированное регулярное выражение в ***preg** в строке string
- **eflags** контролирует способ выполнения; ее значение равно 0 или побитовому ИЛИ одного или более флагов из табл

REG_NOTBOL	Оператор ^ (начало строки) не сопоставляется
REG_NOTEOL	Оператор $ (конец строки) не сопоставляется

- Флаги REG_NEWLINE, REG_NOTBOL и REG_NOTEOL взаимодействуют друг с другом.

regex

- Когда в cflags не включен REG_NEWLINE, символ конца строки действует в качестве обычного символа. С ним может быть сопоставлен метасимвол '.' (любой символ), а также дополненные списки символов ('[^...]'). При этом $ не сопоставляется немедленно с началом вставленного символа новой строки, а ^ не сопоставляется немедленно с его концом.
- Когда в eflags установлен REG_NOTBOL, оператор ^ не соответствует началу строки. Это полезно, когда параметр string является адресом символа в середине сопоставляемого текста.
- Сходным образом, когда в eflags установлен REG_NOTEOL, оператор $ не соответствует концу строки.

regex

Когда в cflags включен REG_NEWLINE, то:

- Символ конца строки не соответствует '.' или дополненному списку символов.
- Оператор ^ всегда соответствует положению непосредственно за вставленным символом конца строки независимо от установки REG_BOL.
- Оператор $ всегда соответствует положению непосредственно перед вставленным символом конца строки независимо от установки REG_EOL.

Когда вы осуществляете построчный ввод/вывод, как в случае с grep, можно не включать REG_NEWLINE в cflags. Если в буфере несколько строк, и каждую из них нужно рассматривать как отдельную, с сопоставлением ^ и $, тогда следует включить REG_NEWLINE.

regex

size_t **regerror**(int errcode, const regex_t *preg, char *errbuf, size_t errbuf_size)

- Преобразует ошибку, возвращенную regcomp() или regexec(), в удобочитаемую строку.

regex

void **regfree**(regex_t *preg)

- Освобождает динамическую память, используемую откомпилированным регулярным выражением в *preg.

regex

regcomp() и regexec() возвращают 0, если они успешны, или определенный код ошибки, если нет.

REG_BADBR	Содержимое '\{...\}' недействительно.
REG_BADPAT	Регулярное выражение недействительно
REG_BADRPT	Символу ?, + или * не предшествует действительное регулярное выражение.
REG_EBRACE	Фигурные скобки ('\{...\}') не сбалансированы
REG_EBRACK	Квадратные скобки ('[...]') не сбалансированы
REG_ECOLLATE	В шаблоне использован недействительный элемент сортировки
REG_ECTYPE	В шаблоне использован недействительный класс символов
REG_EESCAPE	В шаблоне есть завершающий символ \
REG_EPAREN	Группирующие скобки ('(...)' или '\(...\)') не сбалансированы
REG_ERANGE	Конечная точка в диапазоне не действительна
REG_ESPACE	Функции не хватило памяти
REG_ESUBREG	Цифра в '\цифра' недействительна
REG_NOMATCH	Строка не соответствует шаблону

Пример

```
#include <regex.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <malloc.h>

char* paste_env(char *str)
{
    char *env;
    char buf[256];
    regex_t preg;
    regmatch_t pm;
    char *newstr;
    int len;
```

Пример

```
    regcomp( &preg, "\\$\\w\\+\\b", REG_ICASE );
    while ( regexec(&preg, str, 1, &pm, REG_NOTBOL) == 0 )
    {
        memset(buf, '\0', 256);
        strncpy(buf,&str[pm.rm_so]+1,(pm.rm_eo-pm.rm_so-1 < 256)?pm.rm_eo-pm.rm_so-1:254);
        env = getenv(buf); // check for empty
        len = strlen(str)-strlen(buf)+strlen(env);
        newstr=malloc(len);
        memset(newstr, '\0', len);
        strncpy(newstr,str,pm.rm_so);
        strncat(newstr,env,strlen(env));
        strncat(newstr,&str[pm.rm_eo],strlen(str)-pm.rm_eo);
        str = newstr;
    }
    regfree(&preg);

    return newstr;
}
```

Пример

```
int main ()
{
   printf(paste_env("$HOME text $PWD $HOME\n"));
   return 0;
}
```

Дополнительные материалы

- *http://wm-help.net/lib/b/book/2075737573/292*