`getSeq`, v. 0.5: Get Sequence from FASTA File

Bernhard Haubold

November 6, 2018

1 Introduction

`getSeq` matches a regular expression to the headers in a FASTA file and prints out all the sequences with matching headers.

2 Getting Started

`getSeq` was written in C on a computer running Mac OS X; it is intended to run on any UNIX system with a C compiler. However, please contact me at `haubold@evolbio.mpg.de` if you have problems with the program.

- Unpack the program

  ```
  tar -xvzf getSeq_XXX.tgz
  ```

  where `XXX` indicates the version.
- Change into the newly created directory

  ```
  cd getSeq_XXX
  ```

  and list its contents

  ```
  ls
  ```
- Generate `getSeq`

  ```
  make
  ```
- List its options

  ```
  ./getSeq -h
  ```

3 Usage Examples

- Begin by listing the headers of the example data file:

  ```
  grep \> Data/test.fasta
  ```
- Extract single sequence

  ```
  ./getSeq -s 10 Data/test.fasta
  ```
- Extract the complement of a single sequence

  ```
  ./getSeq -c -s 10 Data/test.fasta
  ```
- Extract all sequences whose name contains a “1”

  ```
  ./getSeq -s 1 Data/test.fasta
  ```

4 Change Log

1. v. 0.3 (May 26, 2010)
    - First version distributed.
2. v. 0.4 (February 9, 2011)
    - Fixed user interface.
3. v. 0.5 (November 2018)
    - Fixed bug in `interface.c`.

5 Listings

The following listing documents the central part of the code for `getSeq`.

5.1 The Driver Program: `getSeq.c`

```
/***** getSeq.c ***************************************************
 * Description: get a sequence from a FASTA file
 * Author: Bernhard Haubold, haubold@evolbio.mpg.de
 * File created on Tue May 24 15:55:43 2005.
 *****************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <regex.h>
#include <fcntl.h>
#include <unistd.h>
#include "eprintf.h"
#include "interface.h"

void runAnalysis(Args *args, int fd, regex_t re, char *buf);
char *getLine(int fd, char *buf);

int main(int argc, char *argv[]){
  char *version;
  Args *args;
  regex_t re;
  int i, fd;
  char *buf;

  version = "0.5";
  setprogname2("getSeq");
  args = getArgs(argc, argv);
  if(args->h == 1){
    printUsage(version);
    return 0;
  }else if(args->e == 1){
    printUsage(version);
    return -1;
  }else if(args->v == 1){
    printSplash(version);
    return 0;
  }
  /* compile regex */
  if(regcomp(&re, args->s, REG_EXTENDED) != 0) {
    fprintf(stderr, "%s:␣Error␣compiling␣regular␣expression:␣%s\n", "getSeq
       ", args->s);
    exit(EXIT_FAILURE);
  }
  buf = (char *)emalloc(BUFSIZ*sizeof(char));
  if(args->numInputFiles){
    for(i=0;i<args->numInputFiles;i++){
      fd = open(args->inputFiles[i],O_RDONLY,0);
      runAnalysis(args,fd,re,buf);
      close(fd);
    }
  }else{
    fd = 0;
    runAnalysis(args,fd,re,buf);
  }
  free(args);
  free(progname());

  return 0;
}

void runAnalysis(Args *args, int fd, regex_t re, char *buf){
  int retval, out;
  char *header, *bp;
  int headerLen, headerOpen, headerInd, c;

  headerLen = 1;
  header = (char *)emalloc(headerLen*sizeof(char));
  headerInd = 0;
  headerOpen = 0;
  out = 0;
  while((c = read(fd,buf,BUFSIZ)) > 0){
    for(bp=buf;bp-buf<c;bp++){
      if(*bp == ’>’)
        headerOpen = 1;
      if(headerOpen){
        if(headerInd >= headerLen){
          headerLen *= 2;
          header = (char *)erealloc(header,headerLen*sizeof(char));
        }
        header[headerInd++] = *bp;
        if(*bp == ’\n’){
          headerOpen = 0;
          header[headerInd] = ’\0’;
          headerInd = 0;
          if((retval = regexec(&re,header,0,NULL,0)) == 0){
            if(args->c)
              out = 0;
            else
              out = 1;
```

```
     }else{
      if(args->c)
       out = 1;
      else
       out = 0;
     }
     if(out)
      printf("%s",header);
    }
   }else
    if(out)
     printf("%c",*bp);
  }
 }
}
```