`traverseTree`, v. 0.3: DESCRIPTION

Bernhard Haubold
Max-Planck-Institute for Evolutionary Biology, Plön, Germany

July 11, 2017

1 Introduction

2 Getting Started

`traverseTree` was written in C on a computer running Linux and should work on any standard UNIX system. However, please contact me at `haubold@evolbio.mpg.de` if you have any problems with the program.

- Unpack the program

  ```
  tar -xvzf traverseTree_XXX.tgz
  ```

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

  ```
  cd TraverseTree_XXX
  ```

  and list its contents

  ```
  ls
  ```

- Generate `traverseTree`

  ```
  make
  ```

- List its options

  ```
  ./traverseTree -h
  ```

3 Listings

3.1 The Driver Program `traverseTree.c`

```
/***** traverseTree.c *****************************
 * Description: Read tree in Newick format and
 *   traverse it.
 * Author: Bernhard Haubold, haubold@evolbio.mpg.de
 * Date: Sun Sep 16 20:28:48 2012
 **************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include "interface.h"
#include "eprintf.h"
#include "tree.h"
#include "traverse.h"

void scanFile(FILE *fp, Args *args);
void printHeader();

int main(int argc, char *argv[]){
  int i;
  char *version;
  Args *args;
  FILE *fp;

  version = "0.3";
  setprogname2("traverseTree");
  args = getArgs(argc, argv);
  if(args->v)
    printSplash(version);
  if(args->h || args->e)
    printUsage(version);
  if(args->numInputFiles == 0){
    fp = stdin;
    scanFile(fp, args);
  }else{
    for(i=0;i<args->numInputFiles;i++){
      fp = efopen(args->inputFiles[i],"r");
      scanFile(fp, args);
      fclose(fp);
    }
  }
  free(args);
  free(progname());
  return 0;
}

void scanFile(FILE *fp, Args *args){
  Node *root;

  setBisonFile(fp);
  while((root = parseTree()) != NULL){
    if(strcmp(args->t,"inorder") == 0){
      printf("Inorder␣Traversal\n");
      printHeader();
      printInorder(root);
    }if(strcmp(args->t,"preorder") == 0){
      printf("Preorder␣Traversal\n");
      printHeader();
      printPreorder(root);
    }if(strcmp(args->t,"postorder") == 0){
      printf("Postorder␣Traversal\n");
      printHeader();
      printPostorder(root);
    }
  }
}

void printHeader(){
  printf("ID\tLabel\tParent\tDist.\tType\n");
}
```

3.2 Tree Traversal: `traverse.c`

```
/***** traverse.c *********************************
 * Description:
 * Author: Bernhard Haubold, haubold@evolbio.mpg.de
 * Date: Mon Sep 17 08:14:32 2012
 **************************************************/
#include <stdio.h>
#include <stdlib.h>
#include "tree.h"

void printNode(Node *np);

void printInorder(Node *np){
  Node *child;

  if(np != NULL){
    printInorder(np->left);
    printNode(np);
    if(np->left){
      child = np->left;
      while(child->right){
        printInorder(child->right);
        child = child->right;
      }
      printInorder(child->right);
    }
  }
}

void printPreorder(Node *np){
  Node *child;

  if(np != NULL){
    printNode(np);
    printPreorder(np->left);
    if(np->left){
      child = np->left;
      while(child->right){
        printPreorder(child->right);
        child = child->right;
      }
      printPreorder(child->right);
    }
  }
}

void printPostorder(Node *np){
 Node *child;

 if(np != NULL){
  printPostorder(np->left);
  if(np->left){
   child = np->left;
   while(child->right){
    printPostorder(child->right);
    child = child->right;
   }
   printPostorder(child->right);
  }
  printNode(np);
 }
}

void printNode(Node *np){
 printf("%2d",np->id);
 if(np->label != NULL)
  printf("\t%s\t",np->label);
 if(np->parent)
  printf("%2d",np->parent->id);
 printf("\t%5.1f",np->dist);
 if(np->parent == NULL && np->left != NULL)
  printf("\troot\n");
 else if(np->parent != NULL && np->left == NULL)
  printf("\tleaf\n");
 else if(np->parent == NULL && np->left == NULL)
  printf("\troot/leaf\n");
 else
  printf("\tinternal\n");
}
```

4 Change Log

- Version 0.1, 17 September 2012
  - First version.
- Version 0.2, ???
  - ???
- Version 0.3, 20 January 2013
  - Fixed order of traversal.