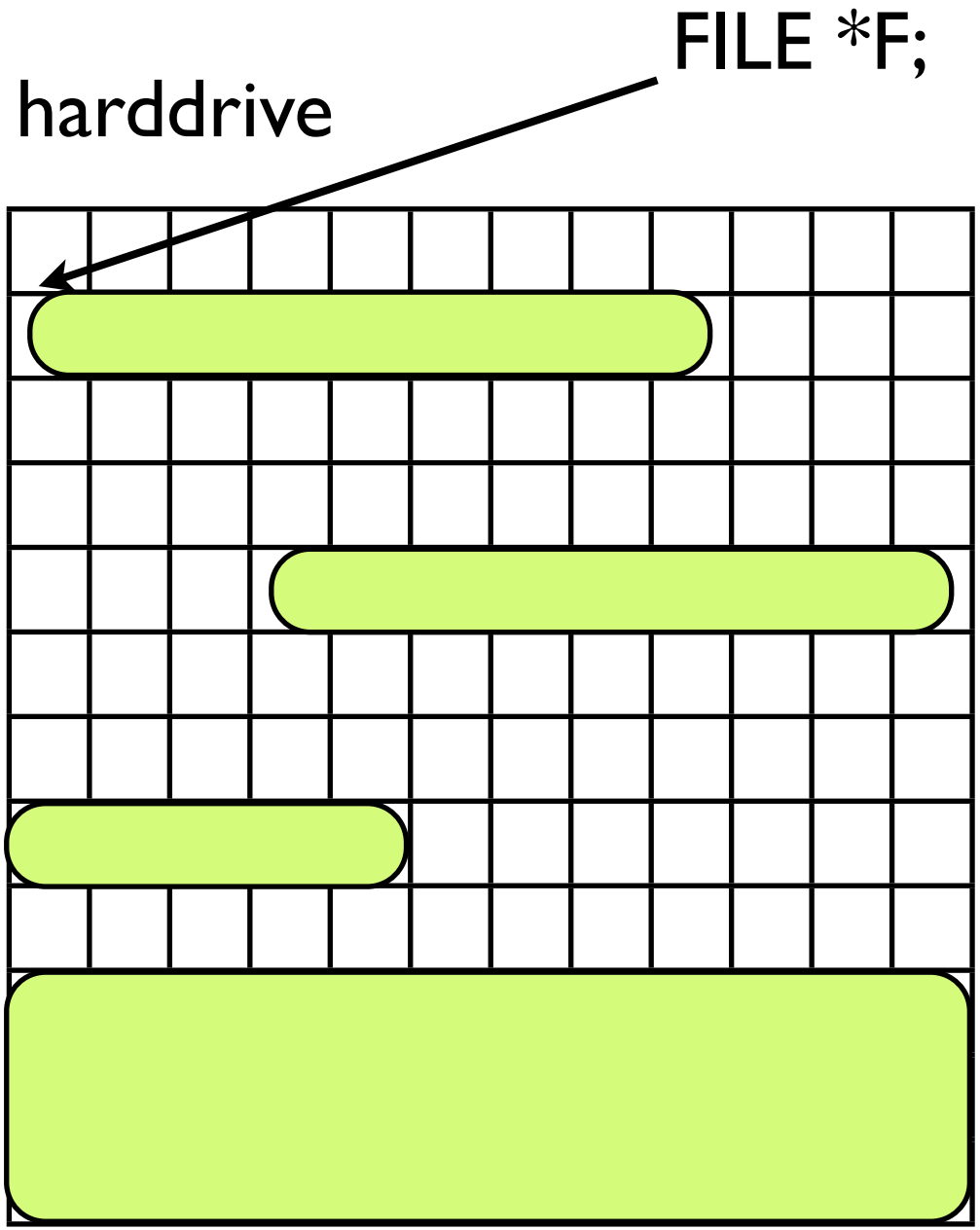
CSE 251 files and more pointers

Arend Hintze

file handling

- you use the “same” commands to read and write to a file than you use for reading and writing to the commandline
- printf->fprintf scanf->fscanf
- you need to know the your file handle -> new type FILE
- you have to open and close the file


FILE *F;
harddrive

read/write head

fopen

FILE *fopen(const char *fname, const char *mode);
The fopen() function opens a file indicated by fname and returns a stream associated with that file. If there is an error, fopen() returns NULL. mode is used to determine how the file will be treated (i.e. for input, output, etc)

Mode	Meaning
"r"	Open a text file for reading
"w"	Create a text file for writing
"a"	Append to a text file
"rb"	Open a binary file for reading
"wb"	Create a binary file for writing
"ab"	Append to a binary file
"r+"	Open a text file for read/write
"w+"	Create a text file for read/write
"a+"	Open a text file for read/write
"rb+"	Open a binary file for read/write
"wb+"	Create a binary file for read/write
"ab+"	Open a binary file for read/write

reading

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

int main(){
    FILE *F;
    int i;
    F=fopen("myText.txt","w+t");
    for(i=0;i<100;i++)
        fprintf(F,"%i\n",i);
    fclose(F);
    return 0;
}
```

writing

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

int main(){
    FILE *F;
    int i,d;
    F=fopen("myText.txt","r+t");
    for(i=0;i<100;i++){
        fscanf(F,"%i\n",&d);
        printf("%i\n",d);
    }
    fclose(F);
    return 0;
}
```

feof

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

int main(){
    FILE *F;
    int d;
    F=fopen("myText.txt","r+t");
    while(!feof(F)){
        fscanf(F,"%i\n",&d);
        printf("%i\n",d);
    }
    fclose(F);
    return 0;
}
```

http://www.elook.org/programming/c/fseek.html

fseek

Syntax:

```
#include <stdio.h>
int fseek( FILE *stream, long offset, int origin );
```

Description:

The function fseek() sets the file position data for the given stream. The origin value should have one of the following values (defined in stdio.h):

Name	Explanation
SEEK_SET	Seek from the start of the file
SEEK_CUR	Seek from the current location
SEEK_END	Seek from the end of the file

fseek() returns zero upon success, non-zero on failure. You can use fseek() to move beyond a file, but not before the beginning. Using fseek() clears the EOF flag associated with that stream.

rewind

Syntax:
#include <stdio.h>
void rewind(FILE *stream);

Description:
The function rewind() moves the file position indicator to the beginning of the specified stream, also clearing the error and EOF flags associated with that stream.

ftell

```
#include <stdio.h>
long ftell( FILE *stream );
```

Description:
The ftell() function returns the current file position for stream, or -1 if an error occurs.

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>
#include <string.h>
int main(){
    FILE *F;
    int d;
    long length;
    F=fopen("myText.txt","r+t");
    fseek(F, 0, SEEK_END);
    length=ftell(F);
    printf("Length of file:%i\n",length);
    rewind(F);
//    fseek(F, length/2, SEEK_CUR);
    while(!feof(F)){
        fscanf(F,"%i\n",&d);
        printf("%i\n",d);
    }
    fclose(F);
    return 0;
}
```

fread

```
#include <stdio.h>
int fread( void *buffer, size_t size, size_t num, FILE *stream );
```

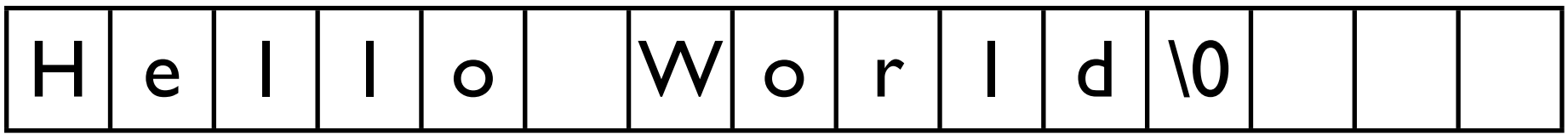
Description:
The function fread() reads num number of objects (where each object is size bytes) and places them into the array pointed to by buffer. The data comes from the given input stream. The return value of the function is the number of things read...use |feof()| or |ferror()| to figure out if an error occurs.

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>
#include <string.h>

int main(){
    FILE *F;
    int d;
    long length;
    char *myFile;
    F=fopen("myFile.txt","r+t");
    fseek(F, 0, SEEK_END);
    length=ftell(F);
    printf("Length of file:%i\n",length);
    rewind(F);
    myFile=(char*)malloc(sizeof(char)*length);
    fread(myFile, sizeof(char), length, F);
    printf("%s",myFile);
    fclose(F);
    return 0;
}
```

todo:

- load a file into a char * (string)
- scan the string and count the number of blanks
- turn that into a function that returns the number of blanks

char * is a string



- char C[6]={'H','e','l','l','o',0};
- char *C="Hello"
- char C[]="Hello";

string.h

- strcpy copies one string into the other
- strcat concatenates strings
- strlen returns the length of a string
- strcmp compares two strings

char *strcpy(char *to, const char *from);
The strcpy() function copies characters in the string from to the string to, including the null termination. The return value is to.

int strcmp(const char *str1, const char *str2);
The function strcmp() compares str1 and str2, then returns:

Return value	Explanation
less than 0	str1 is less than str2
equal to 0	str1 is equal to str2
greater than 0	str1 is greater than str2

size_t strlen(char *str);
The strlen() function returns the length of str (determined by the number of characters before null termination).

char *strcat(char *str1, const char *str2);
The strcat() function concatenates str2 onto the end of str1, and returns str1.

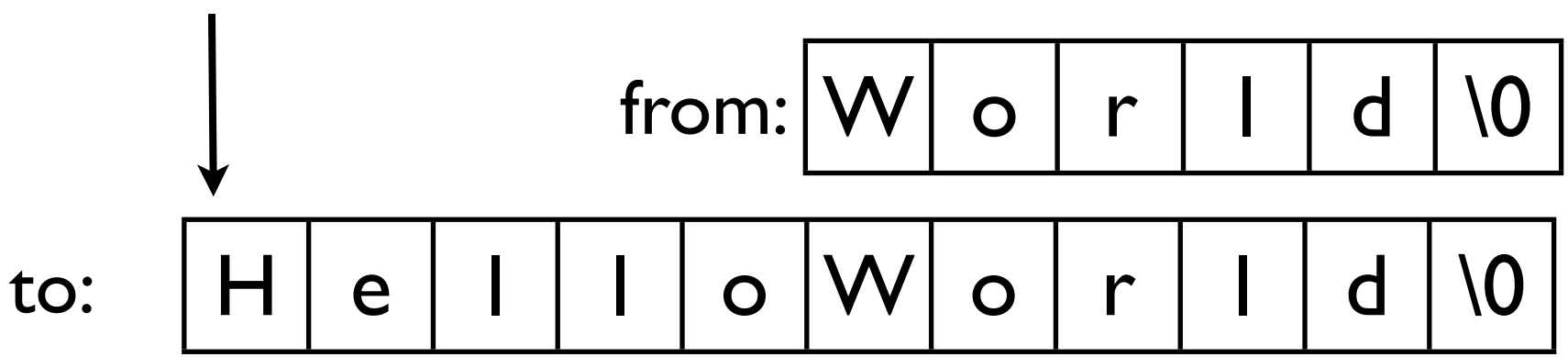
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){
    char *S1="Hello";
    char *S2="World";
    char S3[6];
    char *dest;
    strcpy(S3,S2);
    printf("%s has %i length\n",S1,strlen(S1));
    printf("%s has %i length\n",S3,strlen(S3));
    dest=(char*)malloc(sizeof(char)*(strlen(S1)+strlen(S3)));
    dest[0]=0;
    strcat(dest,S1);
    strcat(dest,S3);
    printf("%s\n",dest);
    printf("%i %i\n",strcmp(S2,S3),strcmp(dest,S1));
    free(dest);
    return 0;
}
```

void *memcpy(void *to, const void *from, size_t count);

Description:
The function memcpy() copies count characters from the array from to the array to. memcpy() returns to. The behavior of memcpy() is undefined if to and from overlap.

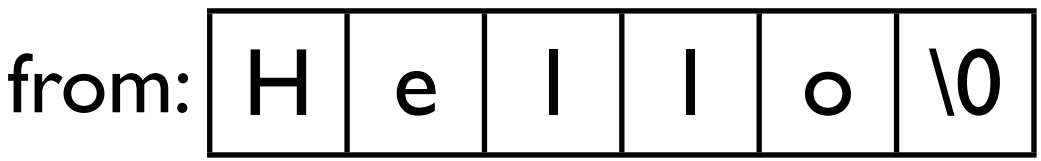
program from before add:

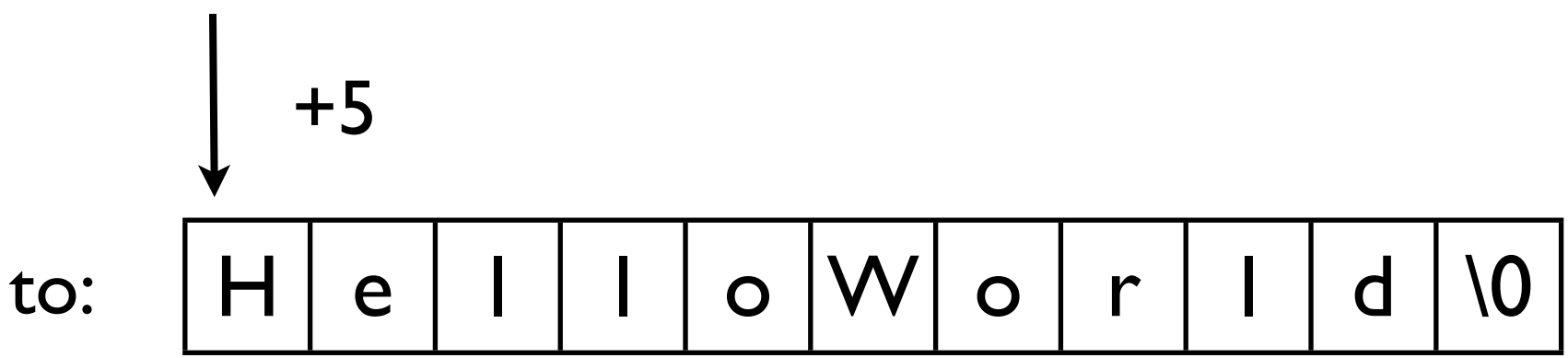
```
    memcpy(dest,S3,5);
    printf("%s\n",dest);
    free(dest);
    return 0;
}
```



program from before add:

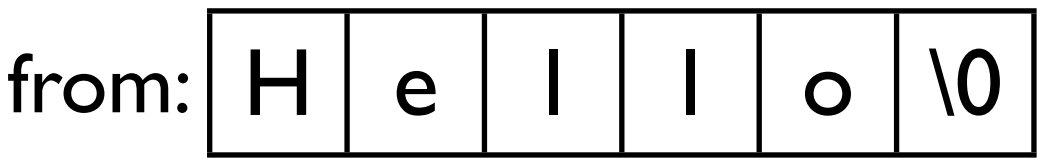
```
    memcpy(dest+5,S1,5);
    printf("%s\n",dest);
    free(dest);
    return 0;
}
```

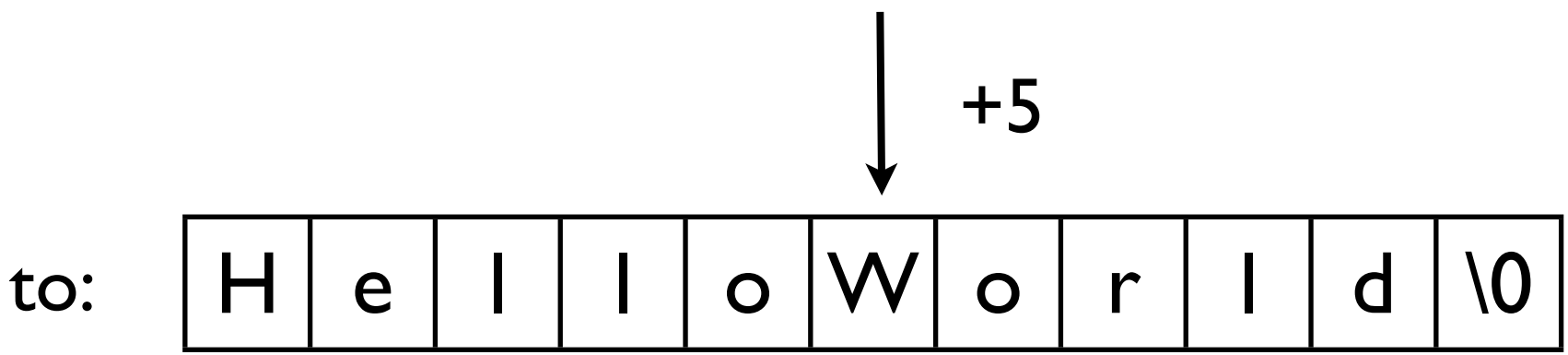




program from before add:

```
    memcpy(dest+5,S1,5);
    printf("%s\n",dest);
    free(dest);
    return 0;
}
```





todo:

- load a file whos filename is specified in argv[1]
- search the file for the occurance of a string specified by argv[2]

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc,const char* argv[]){
    FILE *F;
    char *P,*D;
    int i;
    long length,wordLength;
    F=fopen(argv[1],"r+t");
    fseek(F, 0, SEEK_END);
    length=ftell(F);
    wordLength=strlen(argv[2]);
    rewind(F);
    P=(char*)malloc(sizeof(char)*length);
    D=(char*)malloc(sizeof(char)*(wordLength+1));
    fread(P,sizeof(char),length,F);
    fclose(F);
    printf("%s\n",P);
    for(i=0;i<length-wordLength;i++){
        memcpy(D,P+i,wordLength);
        D[wordLength]=0;
        if(strcmp(D,argv[2])==0)
            printf("Found %s at %i\n",D,i);
    }
     return 0;
}
```

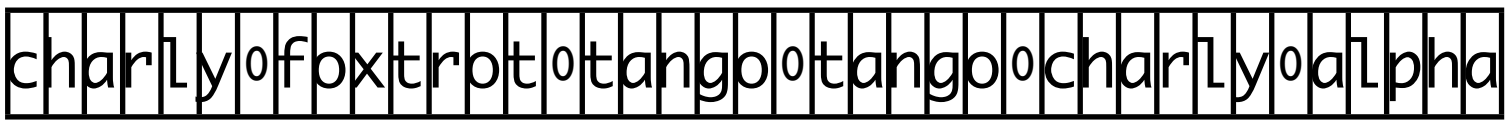
homework

- load a file specified in argv[1]
- count how often each word appears
- example: “charly foxtrot tango tango charly alpha” ->

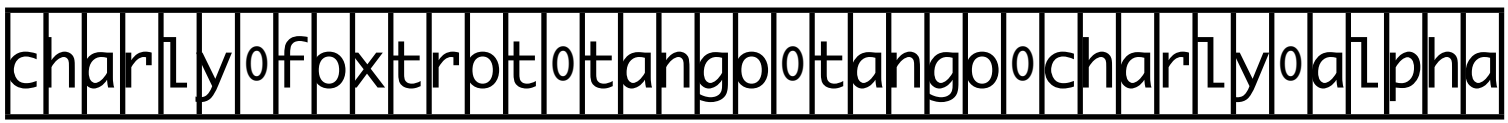
charly 2
foxtrot 1
tango 2
alpha 1

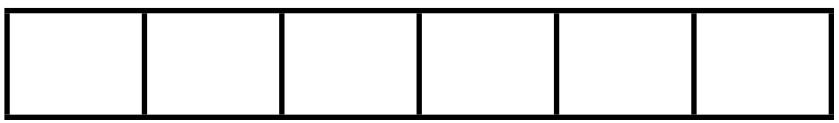

charly foxtrot tango tango charly alpha


charly foxtrot tango tango charly alpha

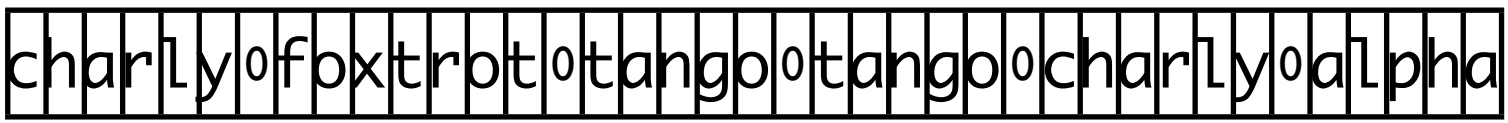

charly0foxtrot0tango0tango0charly0alpha


charly foxtrot tango tango charly alpha


charly0foxtrot0tango0tango0charly0alpha

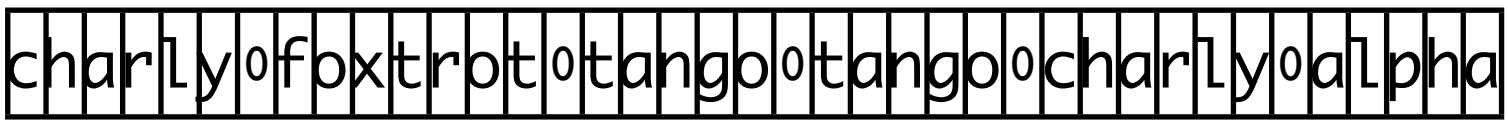



charly foxtrot tango tango charly alpha


charly0foxtrot0tango0tango0charly0alpha

6					


charly foxtrot tango tango charly alpha


charly0foxtrot0tango0tango0charly0alpha

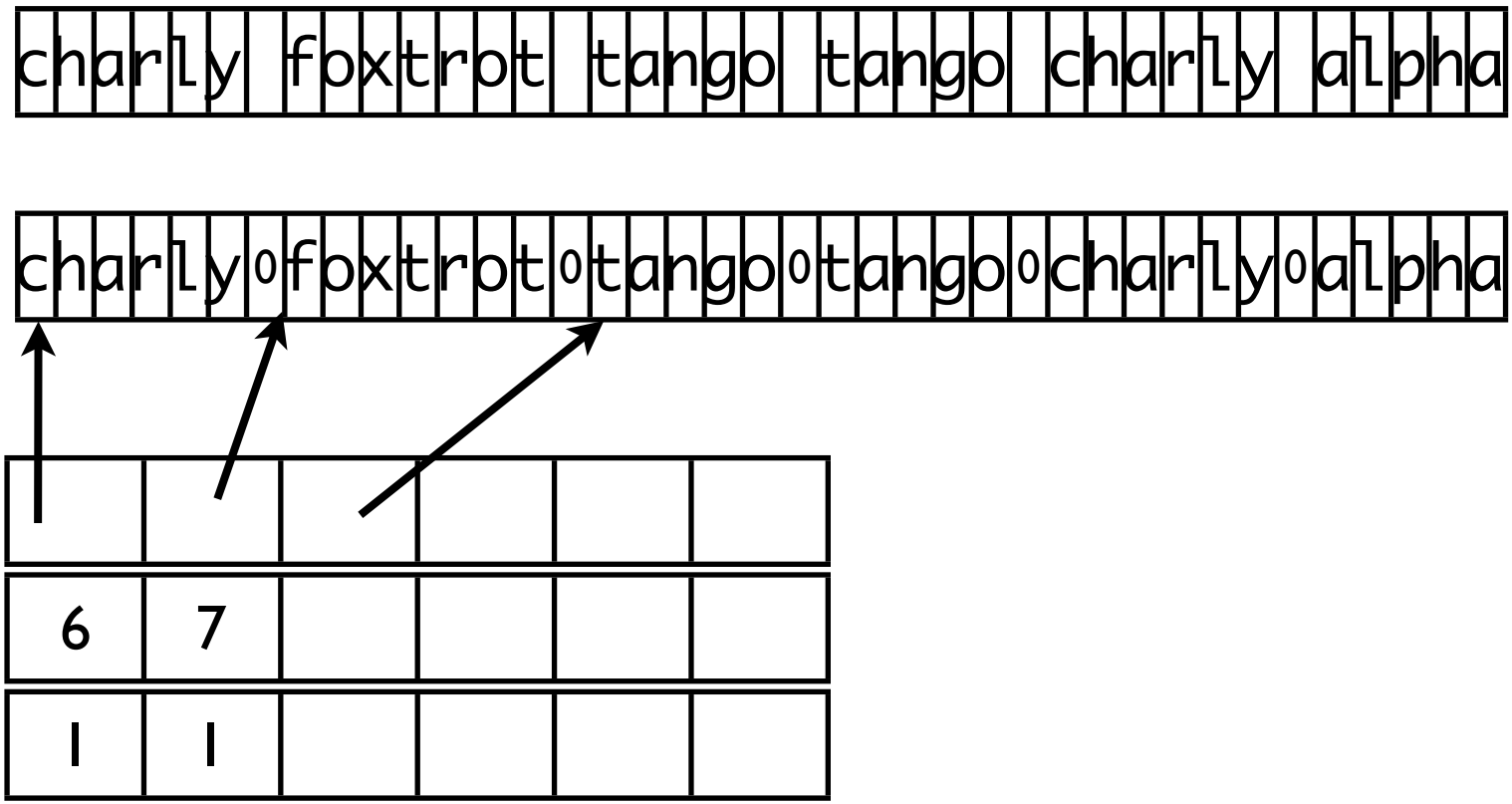
6					
1					

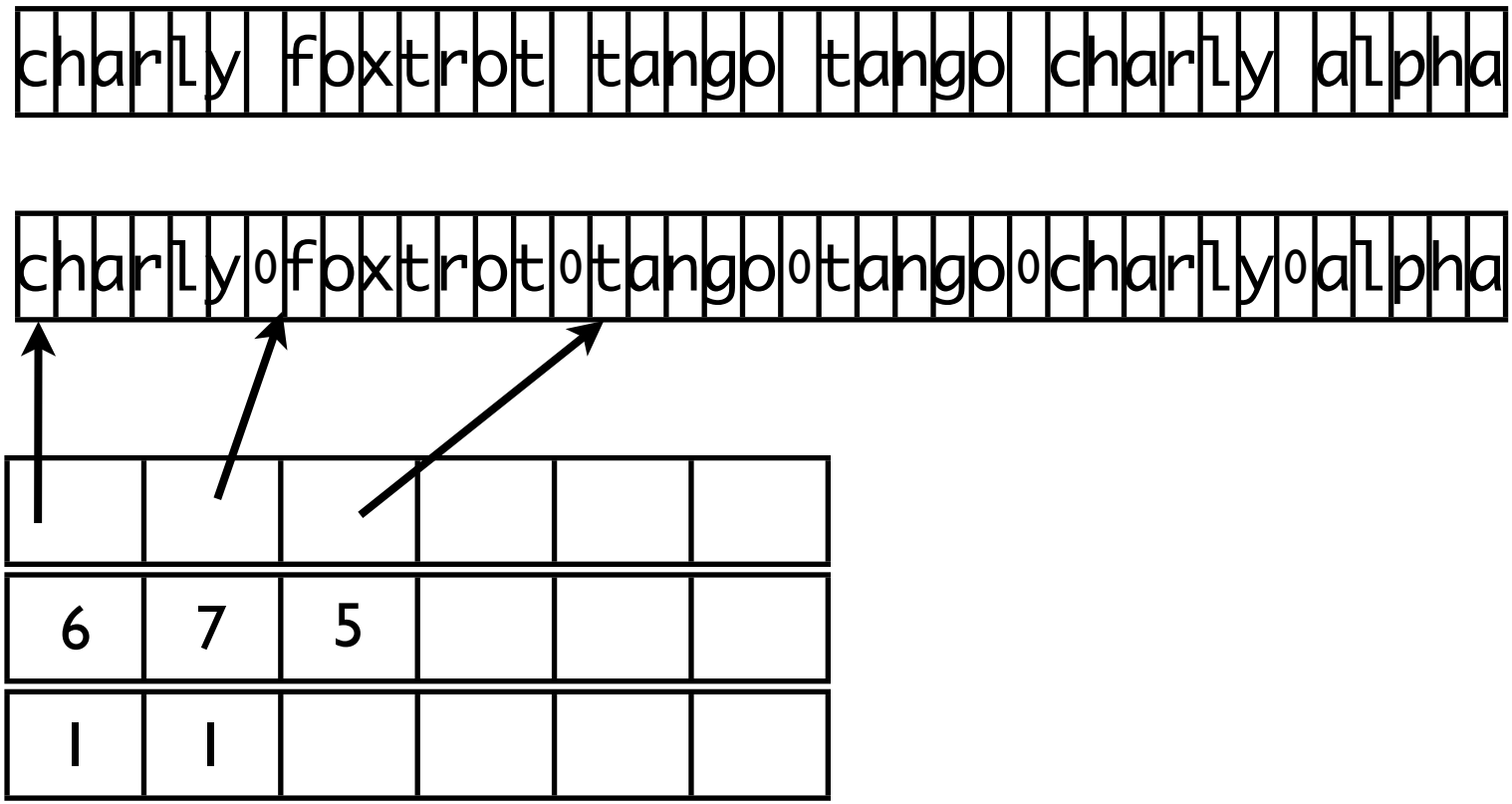

charly foxtrot tango tango charly alpha
charly0foxtrot0tango0tango0charly0alpha
6
1

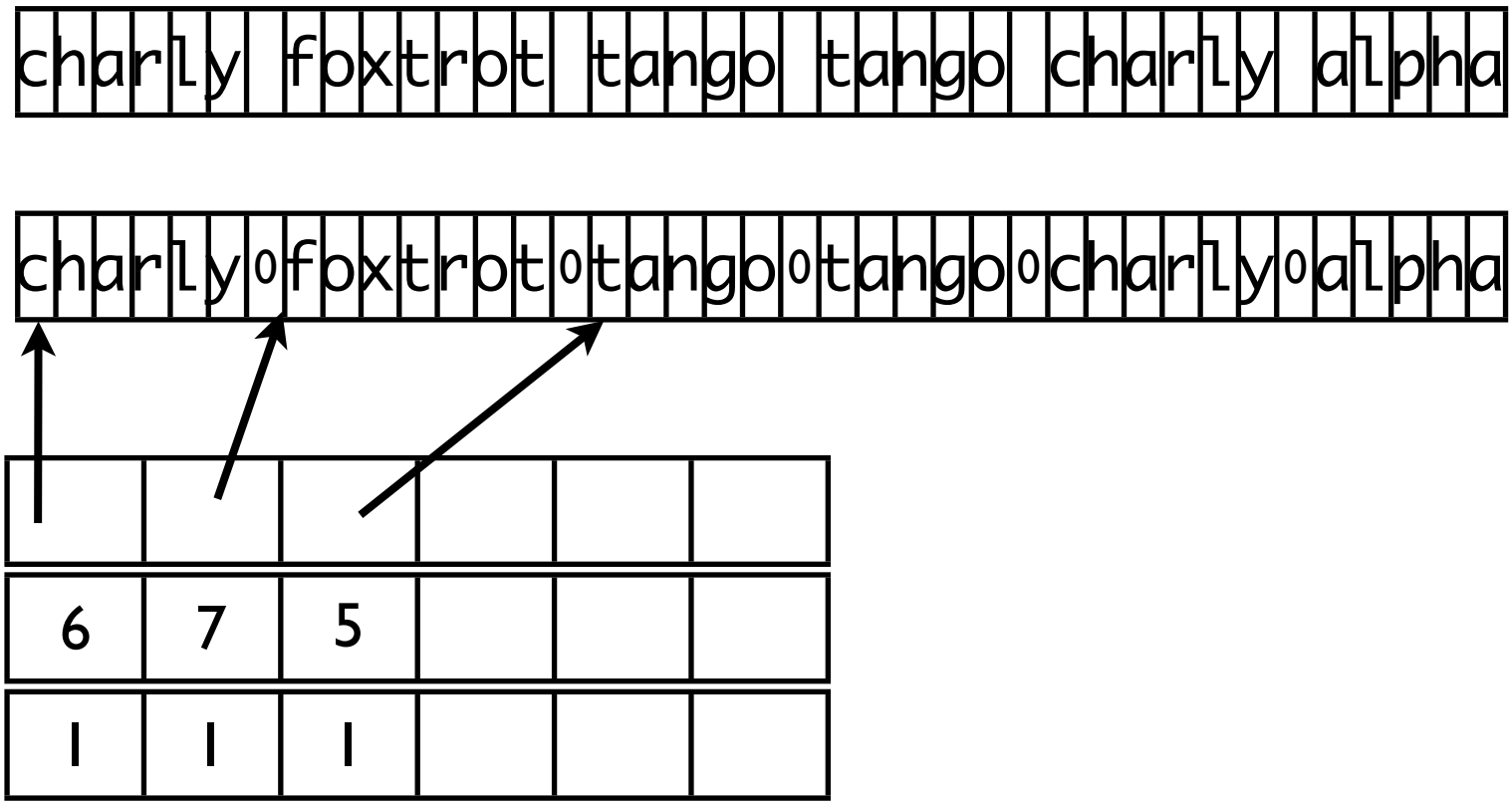

charly foxtrot tango tango charly alpha
charly\0foxtrot\0tango\0tango\0charly\0alpha
6
1

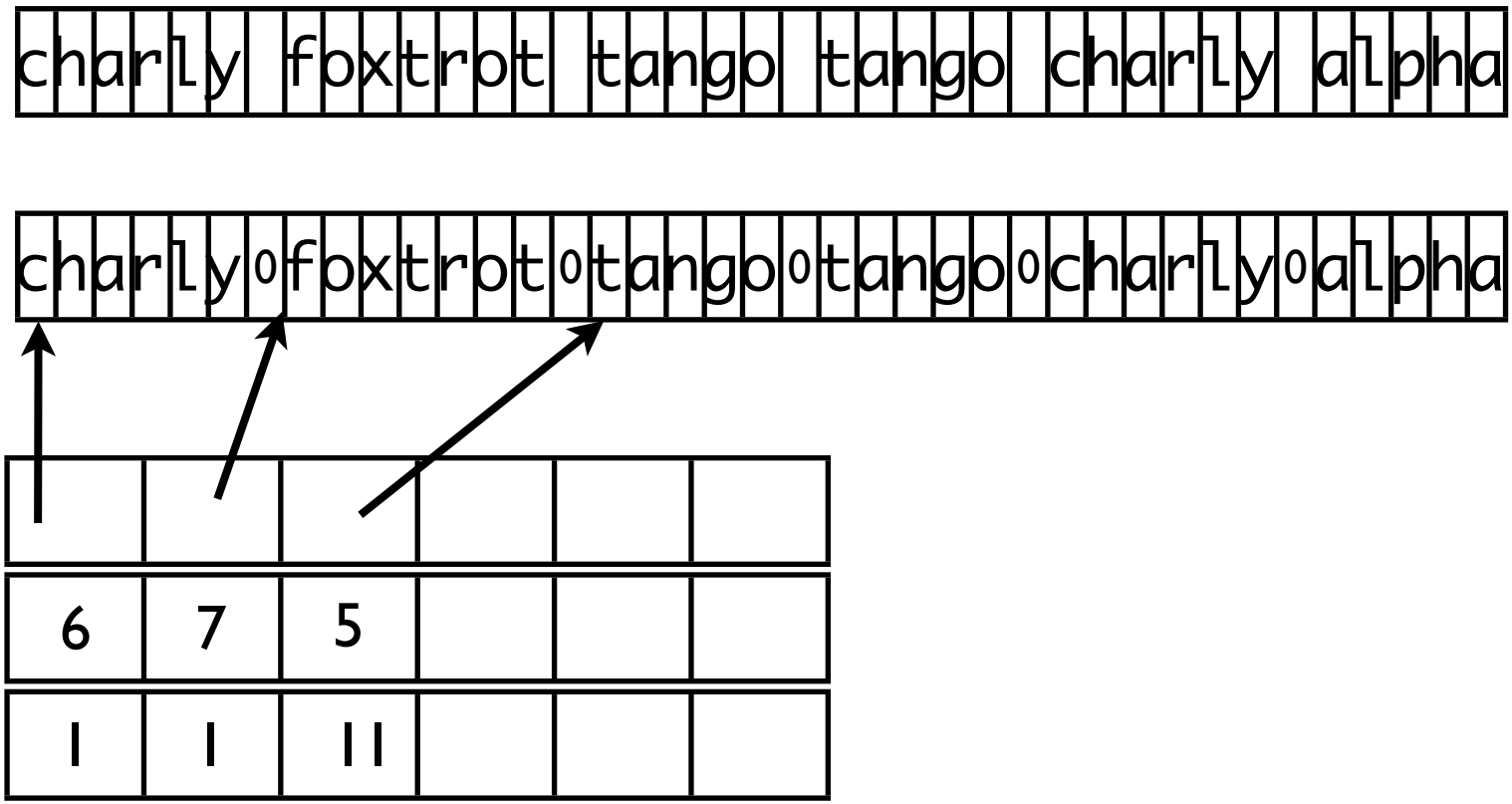

charly foxtrot tango tango charly alpha
charly0foxtrot0tango0tango0charly0alpha
6
7
1

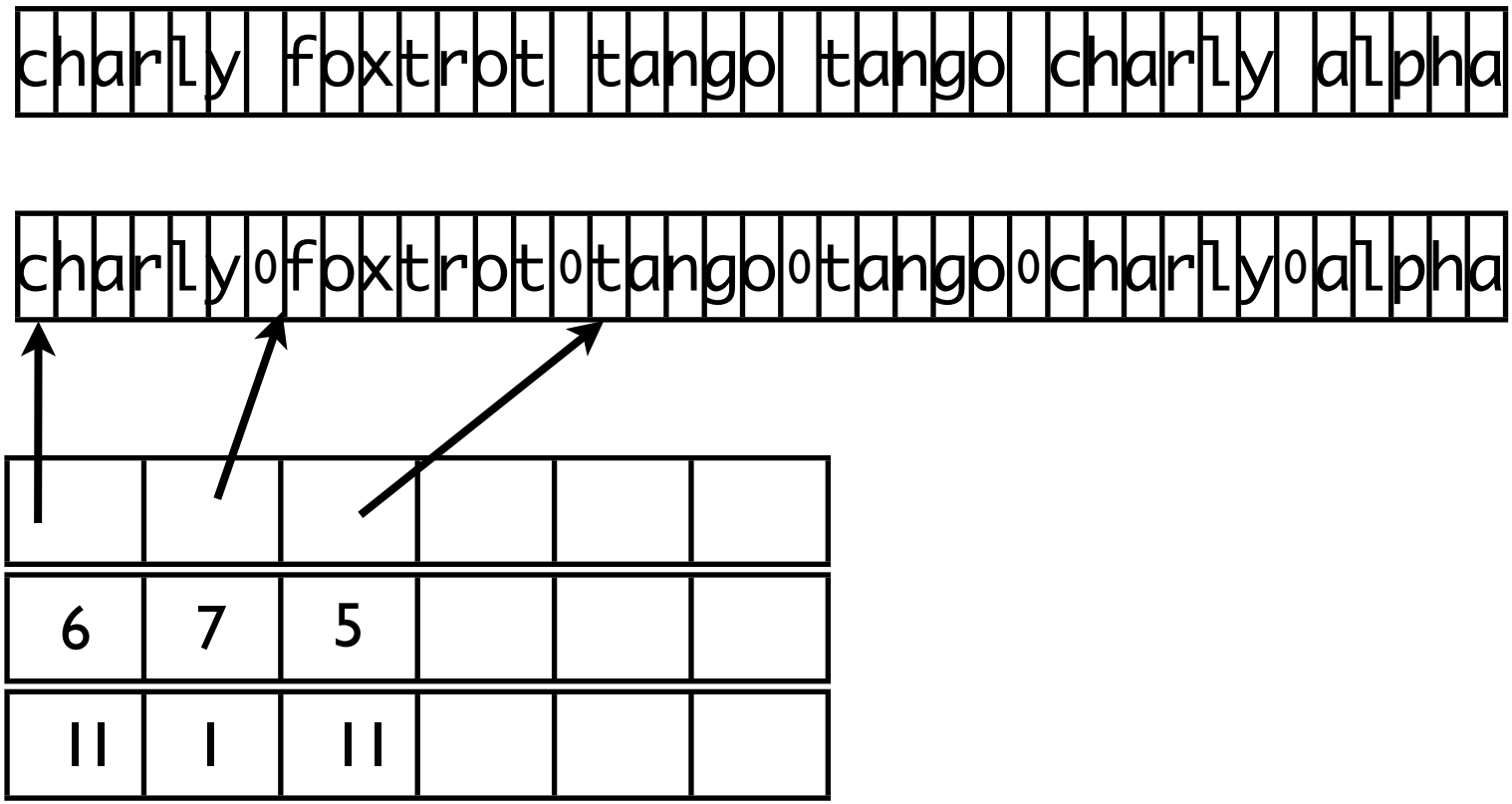

charly foxtrot tango tango charly alpha
charly\0foxtrot\0tango\0tango\0charly\0alpha
6
7
1
1

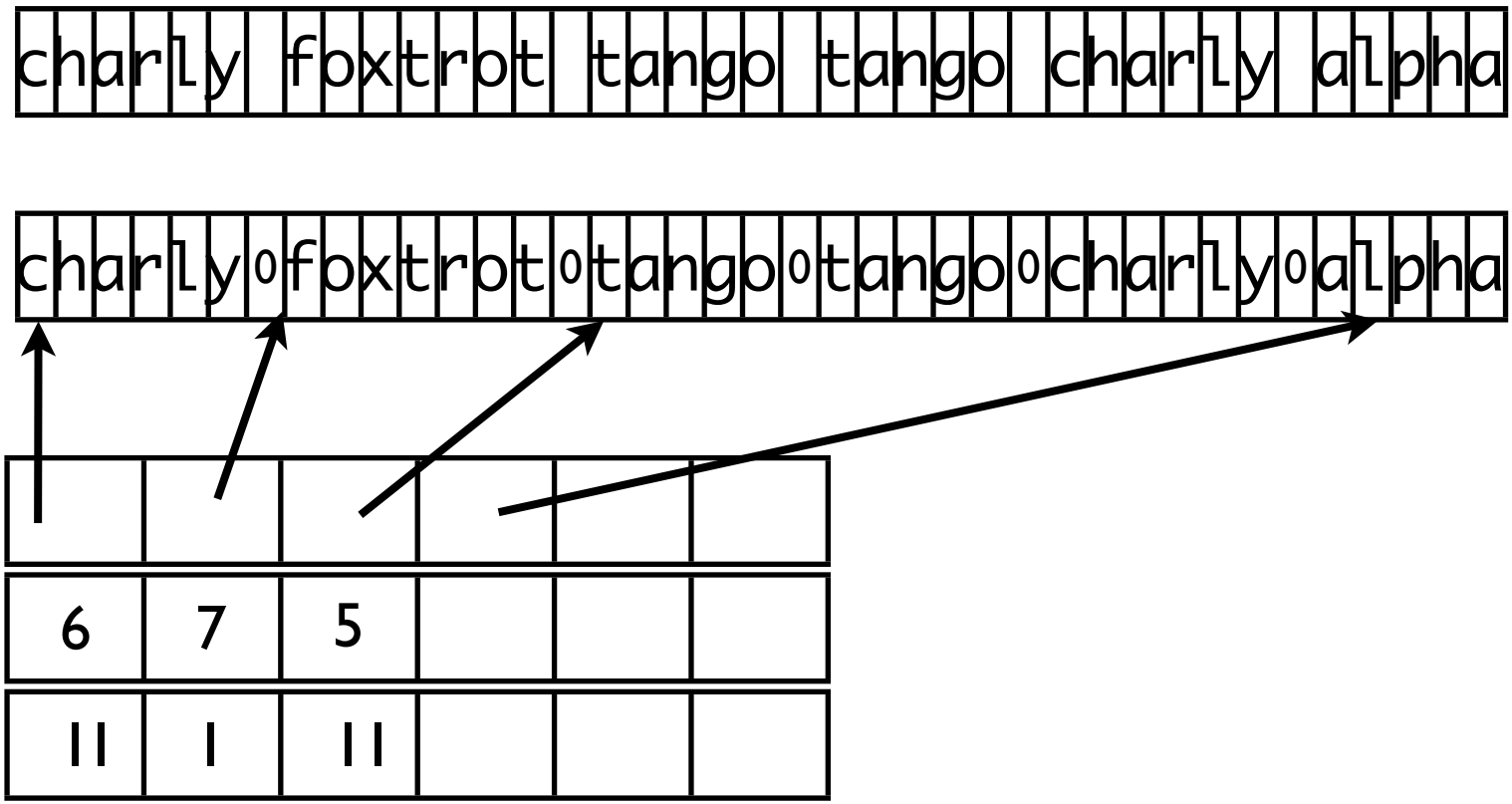

charly foxtrot tango tango charly alpha
charly0foxtrot0tango0tango0charly0alpha
6
7
1
1

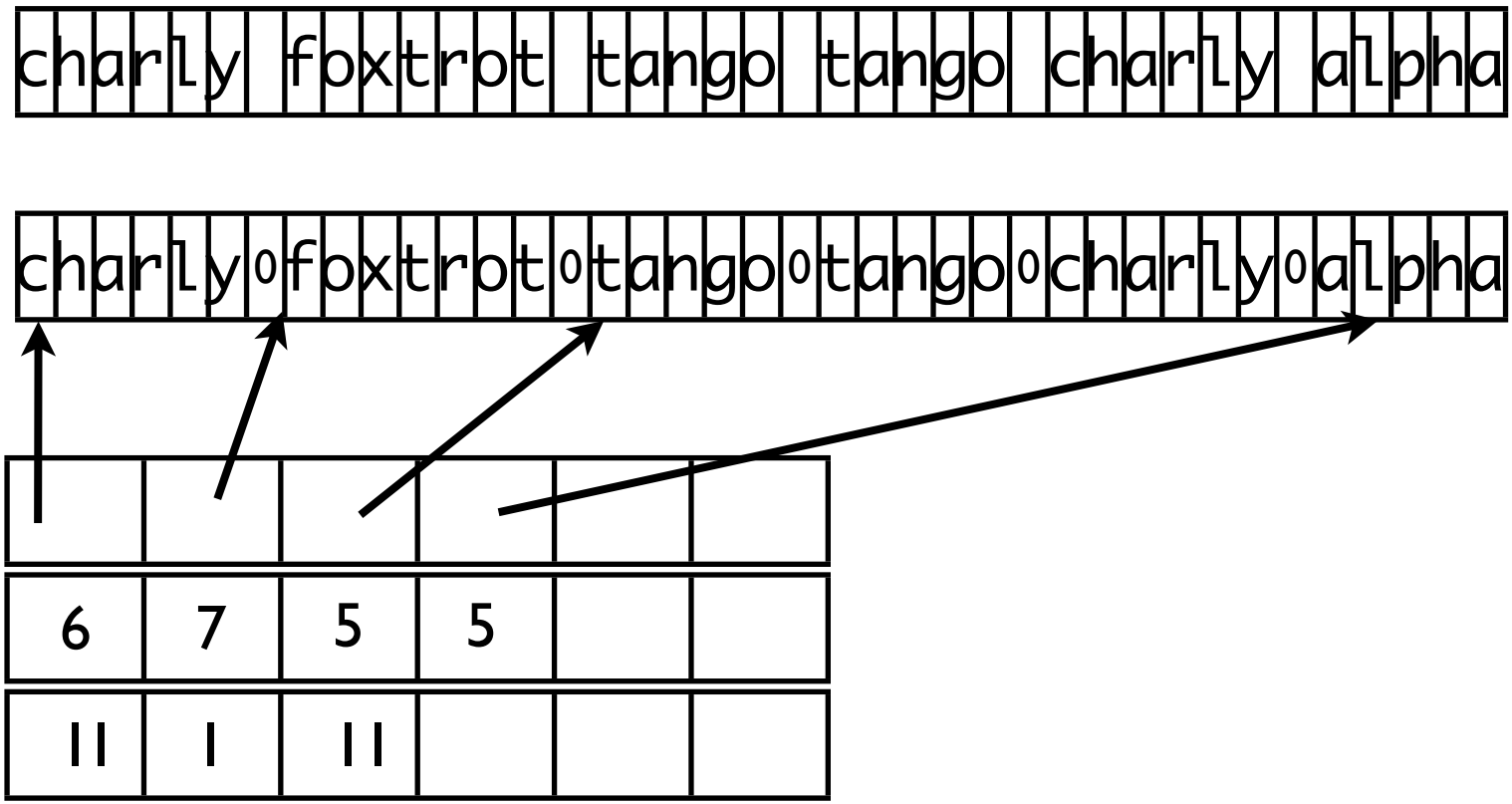

charly foxtrot tango tango charly alpha
charly0foxtrot0tango0tango0charly0alpha
6
7
5
1
1

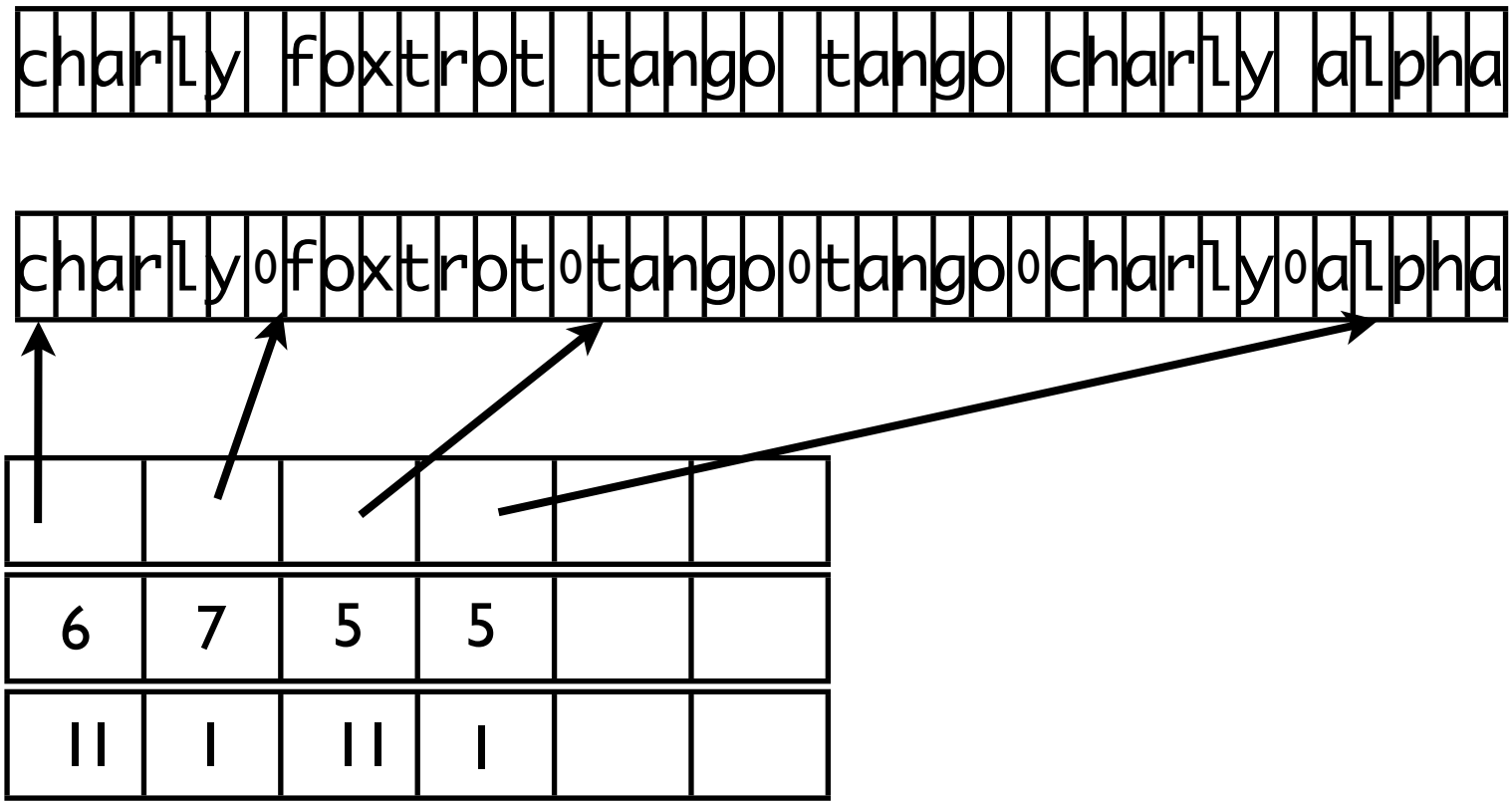

charly foxtrot tango tango charly alpha
charly0foxtrot0tango0tango0charly0alpha
6
7
5
1
1
1


charly foxtrot tango tango charly alpha
charly0foxtrot0tango0tango0charly0alpha
6
7
5
1
1
11


charly foxtrot tango tango charly alpha
charly0foxtrot0tango0tango0charly0alpha
6
7
5
II
I
II


charly foxtrot tango tango charly alpha
charly0foxtrot0tango0tango0charly0alpha
6
7
5
II
I
II


charly foxtrot tango tango charly alpha
charly0foxtrot0tango0tango0charly0alpha
6 7 5 5
II I II


charly foxtrot tango tango charly alpha
charly0foxtrot0tango0tango0charly0alpha
6 7 5 5
II I II I