

CSI 333 – Programming at the Hardware/Software Interface
Program Example for Command Line Arguments

Handout 6.6

This program is functionally the same as the vowel filtering program of Handout 6.3. The key difference is that instead of hardcoding the names of the input and output files (as in Handout 6.3), this version uses command line arguments to get the input and output file names. Assuming that the executable version of this program is named `vfilter`, the form of Unix command needed to run this program is

```
vfilter infile outfile
```

where the arguments `infile` and `outfile` specify the names of the input and output files.

```
#include <stdio.h>
#include <ctype.h>

/* Symbolic constants for command line. */
#define NUMARG 3
#define IN_FILE_ARG 1
#define OUT_FILE_ARG 2

/* Vowel filtering program with command line arguments. */

int main(int argc, char *argv[]) {
    FILE *finp; /* File pointer for input file. */
    FILE *fout; /* File pointer for output file. */

    int c; /* To read chars from input file. */

    int is_vowel(int); /* Prototype of function used. */

    /* First make sure that the command line has exactly */
    /* three arguments (including the command name). */

    if (argc != NUMARG) {
        printf("Usage: vfilter infile outfile\n");
        exit(1);
    }

    /* Strings argv[1] and argv[2] are assumed to specify */
    /* the names of the input and output files respectively. */

    /* Open the input file for reading. */

    if ((finp = fopen(argv[IN_FILE_ARG], "r")) == NULL) { /* Open failed. */
        printf("Could not open file %s for reading.\n", argv[IN_FILE_ARG]);
        exit(1);
    }
}
```

```

/* Open the output file for writing. */

if ((fout = fopen(argv[OUT_FILE_ARG], "w")) == NULL) { /* Open failed. */
    printf("Could not open file %s for writing.\n", argv[OUT_FILE_ARG]);
    exit(1);
}

/* Both input and output files opened successfully. Read */
/* characters from input file, filter out vowels and */
/* write the other characters to the output file. */

while ((c = getc(finp)) != EOF) { /* Note the use of getc. */
    if (!is_vowel(c))
        putchar(c, fout); /* Note the use of putchar. The file pointer is */
                          /* the second argument. */
} /* End of while. */

if (fclose(finp) != EOF) { /* Error in closing input file */
    printf("Error in closing file input.dat.\n");
}

if (fclose(fout) != EOF) { /* Error in closing output file */
    printf("Error in closing file output.dat.\n");
}

return 0;
} /* End of main. */

int is_vowel(int c) {

    /* Returns 1 if parameter c is a vowel (upper or lower case) and 0 otherwise. */

    /* First, convert c to lower case, if it is an upper case letter. */
    c = tolower(c);

    /* Now, we need to check only whether c is a lower case vowel. */
    if ((c == 'a') || (c == 'e') || (c == 'i') || (c == 'o') || (c == 'u'))
        return 1;
    else return 0;
} /* End of is_vowel. */

```