Exercises-Cmdline

Part One

outargs.c
Write a program that simply outputs the number of command line arguments and the arguments one at a time with the argument number.

opts1.c
Write a program that takes three options: \(-j\) \(-k\) \(-m\) They can appear in any order. After processing the arguments it should output a message indicating which one(s) it saw. It should give a message if an illegal option or an argument that is not an option is found and continue processing arguments.
Keep this program. You will be modifying it several times in this exercise set.

opts2.c
Beginning with \texttt{opts1.c}, add two options that are mutually exclusive: \(-a\) \(-b\). Only one of them may be specified. If both are seen, you should give an error message on receipt of the second, and continue processing arguments. Make some decision about which argument has precedence over the other and indicate this in the error message.
Try to break \texttt{opts2.c} by giving it different combinations of options and arguments.
Keep this program. You will be modifying it several times in this assignment.

opts3.c
This program takes an optional two-argument pair \(-n\ name\) and an optional \texttt{filename}. If both are present, the filename must be last. If you can enhance \texttt{opts2.c} with these additional options, do so. Otherwise, write \texttt{opts3.c} as a separate program.
\texttt{opts3.c} should detect illegal or redundant options whether you create it as a separate program or using \texttt{opts2.c}. Try to break it by giving it different combinations of options.

display.c
Last you will write the code to process command-line arguments for a program that has a combination of the different argument types above. Here is the synopsis:
\begin{verbatim}
   display \{-f|\-r\} \{-h\} \{-o output\} input
\end{verbatim}
Here, the options \(-f\) and \(-r\) are mutually exclusive, but one of them must be specified. the \(-h\) and \(-o\) \texttt{output} options are optional, and the \texttt{input} argument is required. The argument must follow all the options.
Write code to process the arguments and diagnose errors and inconsistencies. After processing the arguments, report on the status of each option.
Keep this program. You will be modifying it in a later exercise set.

Part Two

translate
This program implements part of a useful standard unix command named \texttt{tr}. \texttt{translate} simply reads standard input and copies it to standard output after modifying it as follows:
\begin{itemize}
\item \texttt{translate \-d 'c'}
   Copy each character except the character \texttt{c} from standard input to standard output.
\item \texttt{translate \-s 'c'}
   If you see adjacent instances of \texttt{c} on standard input, only copy one instance to standard output.
   ("squeeze" mode).
\end{itemize}
Note: In the examples above, the quotes are not part of the argument when your program accesses them. The last argument in the argument list is a string containing a single character.
Answers and Hints

Part One

goargs.c
Simply use an integer counter and a while loop. Initialize the integer counter to zero, and
while argc>0 output the next argument using argv, decrement argc, and increment argv.

go1.c
Simply use a while loop again, with a nested if. Use three flag variables to indicate whether the option
has been seen. Initialize the flag variables to zero and set them when the option is seen. Make sure you
give an error message if an argument doesn't start with '-'
After the loop, examine each flag variable and output a message if the corresponding option was seen.

go2.c
Add clauses to the if/then/else for -a and -b. In these clauses, give an error message if the
complementary option has already been seen. Whether you take the first one as the correct one or
override it with the second is up-to-you. (If you decide to do the latter, don't forget to turn off the first
option when you turn on the second!)

go3.c
There are several things to check in the implementation of this program:
• don't just assume there is an argument following the -n. The user may be confused. Attempting to
access a non-existent argument may cause a fault!
• Don't forget to increment argv and decrement argc for the name part of the -n name option pair.
  (The update of these variables due to the -n option itself is probably built into the loop, but the update
  necessary for the name part isn't)

display.c
The solution to this program is in the cmdline directory on the public work area. It has code to process
each of the types of arguments in gopts[1-3].c, so you can use it for hints for each of those programs.

Part Two

go1.c
To see how go1.c is supposed to work, try the following experiments at the command line in Unix:
go2.c
echo "hellohellohellothere,Sal11111111lo there,Sal11111111ly" | tr -s 'l'
go3.c
echo "ha#ba#dashery##y" | tr -d '#'

You would be silly to implement this using nget or fgets. Use getchar or fgetc and putchar or
fputc. Continue until the character read is EOF (from <stdio.h>)
There is no solution for this program.