

Problem Sheet #2

Problem 2.1: *xargs* - execute a programs with constructed argument lists (8+2 = 10 points)

Write a C program called *xargs*, which is a simplified version of the Unix *xargs* utility.

- a) Your *xargs* program reads lines from the standard input and constructs argument lists for a command to be executed. If more lines are available than can fit on the argument list, then additional commands are executed with the arguments that did not fit into the argument list of the first command. Your program continues constructing argument lists and executing commands until the end of the standard input has been reached. The command to execute is specified as part of the *xargs* arguments or if none are provided, then `/bin/echo` is used. Your program should support the following options:
- n the (maximum) number of input lines added to the constructed argument lists
 - t show the argument list (on stderr) before the command is executed
- b) Implement the following additional option:
- j the (maximum) number of processes (jobs) executed concurrently (default is 1, which means processes are executed sequentially)

Make sure your program properly handles all possible runtime errors and that it returns an error status to its parent process (usually the shell) in case a runtime error occurred.

Your program must use the `fork()`, `execvp()`, and `waitpid()` system calls. Use the `getopt()` function of the C library for the command line option parsing. You may want to use the `getline()` library function for reading the lines from standard input.

Below are some example invocations:

```
$ echo "hello world" | xargs
hello world
```

```
$ seq 0 10 | xargs
0 1 2 3 4 5 6 7 8 9 10
```

```
$ seq 0 10 | xargs -t
/bin/echo 0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10
```

```
$ seq 0 10 | xargs -n 3
0 1 2
3 4 5
6 7 8
9 10
```

```
$ seq 0 10 | xargs -n 3 -t
/bin/echo 0 1 2
0 1 2
/bin/echo 3 4 5
3 4 5
/bin/echo 6 7 8
6 7 8
/bin/echo 9 10
9 10
```

```
$ seq 1 4 | xargs -t -n 1 printf "foo-%02d\n"  
printf foo-%02d\n 1  
foo-01  
printf foo-%02d\n 2  
foo-02  
printf foo-%02d\n 3  
foo-03  
printf foo-%02d\n 4  
foo-04
```