Midterm#2

- Write a program, say you named it <prog>, which receives as parameters on the command line:
  - An integer
  - One or more paths (either relative or absolute)

  e.g.:

  prog 5 ./relpath1 /abspath2 relpath2

Midterm#2

- The program will then start scanning the sub-trees of the filesystem, starting at each of the paths given as parameters, collecting sizes for all regular files found in each sub-directory.
- Then, it will report the <N> files (the integer parameter) which have the largest size (along with their usage of blocks and complete paths).
- Program should check for errors on input parameters (e.g. all paths must exist and refer to directories, non-negative numbers of items, etc..).

Midterm#2

- You can use the provided shell script (“mkfiles”) to generate a scenario to check the output of your program.
- Assuming that you have both “mkfiles” and “prog” in your home directory, the output of an execution should be similar (although formatting may be different) to that reported in the following slide.

Midterm#2

```
$ cd /tmp
$ bash ~/mkfiles
$ ~/prog 7 $PWD/Dir1 ./Dir4 ./Dir5 $PWD/Dir2 ./Dir3
```

PATH	BLOCKS	SIZE
./Dir3/Sub1/File2	64	32610
/tmp/Dir1/Sub1/Sub2/Sub3/Sub4/Sub5/File2	64	32035
/tmp/Dir2/Sub1/Sub2/Sub3/Sub4/Sub5/File5	64	31488
./Dir4/Sub1/Sub2/Sub3/Sub4/File5	64	31144
./Dir4/Sub1/Sub2/Sub3/Sub4/Sub5/File2	64	30471
./Dir5/Sub1/Sub2/Sub3/Sub4/Sub5/File1	64	30413
./Dir5/Sub1/Sub2/Sub3/Sub4/Sub5/File3	64	30358

Midterm#2

- You can use syscalls related to:
  - Processes (`fork`,`exec`,`wait`, ..).
  - IPC (`sigaction`, `kill`, `pipe`, `dup`, …).
  - Filesystem (`opendir`, `readdir`, `open`, `read`, `stat`, ..).
- Please **don't use**:
  - The `system()` sycall, or other smart tricks to execute shell commands.
  - Stream I/O functions.

Midterm#2

- Hint:
  - Use (if needed) `fork/exec/pipe` with unix commands (e.g. `sort`)