1. Systems

```
$ ls -l big
-rw-r--r-- 1 eggert faculty 9000000000000000000 Sep 21 11:31 b
ig
$ grep x big
$ time grep x big
real 0m0.009s
```

The grep command here analyzes roughly 10^{21} bytes/second...which is unrealistically fast!
How is this possible?
..... big is a sparse file! (generated from $ truncate ... big)
    => grep "cheated" by knowing the file was empty
However...

```
$ grep -r
```

    does not skip sparse files, so it can be stalled...and it is used by the NSA!

Clearly, Operating System choices are important.

How do we define an Operating System?
- ~ American Heritage; 4th Edition (2000)
  - Software designed to **<u>control</u>** the hardware of a **<u>specific</u>** data processing system input & output to allow users & applications to make use
  - ~ This definition wouldn't include Linux, since it works on most hardware!

- ~ Encarta (2007)
  - The master control program in a computer

- ~ Wikipedia v. 917297650
  - System software that manages computer hardware & software resources & provides common services for computer programs

The definition has moved from control to user-environment interaction facilitation
    ~ we can get our goals from this information!

What are the goals of an Operating System?

External Goals:

- protection (from hackers and bugs)
- robustness (in unforeseen circumstances)
- utilization (time wise; always working)
- performance (time/memory/energy)

Internal Goals:

- simplicity (complexity = cost)
- flexibility (easy to mutate)

BUT systems have their downsides...

A. TRADEOFFS

- **Waterbed Effect ~** fixing a problem causes another unrelated one
- **Binary Classification ~** when we must classify binary items by proxy
    - less errors of one type means more of another!
- Say we want to sort 6 million records of 1024 bytes
    - ~ 6 million records take ~6GB! We need to be more space efficient!
- SO we use a pointer array!
    - ~copying is now 2^7 times faster
    - BUT we sacrificed simplicity! (we must edit our software to use pointers)
- The moral of the story is NOTHING IS FREE

B. INCOMMENSURATE SCALING

- Not all parts of a system grow at the same rate.
- There are two common types:
    - Economies of Scale: Big = Fast (think: factory)
    - Diseconomies of Scale: Small = Fast (think: STAR network)

C. EMERGENT PROPERTIES

- Systems sometimes have properties that are not present in any individual members
- ex) Napster and Torrents
    - UCLA received a network speed boost
    - This resulted in an increase in music torrenting!

D. PROPAGATION OF EFFECTS

- ex) Shift-JIS
  - str: ab片c = a | b | / | ... | c
    - Japanese characters had to be modeled as two byte characters
      - BUT some filenames failed since the arbitrary bits could look like a slash!
- We cannot always predict the consequences of a chance in all areas of the system

More broadly, systems introduce a lot of complexity

Complexity is hard to define, so we look for benchmarks:

i) a large # of components

ii) a large # of interconnections

iii) a large # of irregularities

iv) a team of designers, implementers, or maintainers

v) a long Kolmogorov Complexity

Kolmogorov Complexity ≔ the length of the shortest description

Not all signs are necessary for a system to be complex!

We can generalize our problems to a few major Operating System goals:

- virtualization
- concurrency
- persistence

(and if it were up to eggert)

- evolution
- flexibility

~Notice that these are contradictory...especially eggert's!

But why even use an OS?

LETS try to make an application without one!