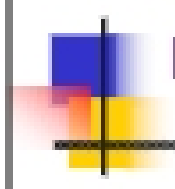
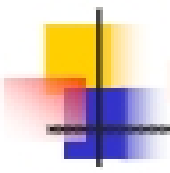


Multics: Dynamic Linking

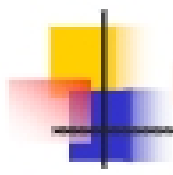
Arvind Krishnamurthy

Spring 2004



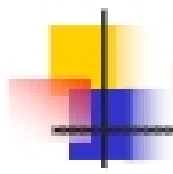
Multics

- Today: look at Mutics VM
- CTSS: probably the first time-sharing system (1959-1965)
  - "Compatible Interactive Time Sharing System"
  - Introduced:
    - working online
    - storing information online
- No protection, off-the-shelf hardware
- 4 consoles running at 110 baud attached to an IBM machine
  - Two tape drives/user, swapped programs and data
- Introduced interactive debugging, editors, command-line processors
- Simple, few key ideas, throw out irrelevant, highly successful



Multics (MIT/Bell/GE)

- Multics: "second system effect"
  - Huge, complicated, tough to debug, terrible performance
- Designed around 1965
  - New hardware, new OS, new programming language
  - Multiple processes, separate address spaces, segmentation with paging
  - Take an interesting idea to the extreme (good research direction!)
    - Extreme sharing
    - Support sharing & dynamic linking
- Unix: third system
  - Understand the limits from the second system, step back, choose with taste, pick some key ideas



Key Ideas in Multics VM

- Combine virtual memory & file systems
  - Two ways to refer to data: (segment number, offset) and (file name, offset); segment is stored on disk or memory
  - Kind of like "mmap" for all data
- Fine-grain sharing
  - Multics took sharing to the extreme
  - Sharing at the level of segments
  - Process = many segments (data or code)
  - Individual library packages are shared; different subsets of processes share different libraries
- Dynamic linking
  - Segments can be "made known" at runtime
  - Share information and upgrade incrementally
- Autonomy (independent address space per. process)
  - Two different libraries might be at different addresses on different processes
  - Need that if we have to support fine-grain sharing

Static Linking Review

```
int y;
extern int z;

int foo() {
        y = 1;
        z = 2;
}
```

[foo.c]

```
000: move xxx, r1
004: store 1, (r1)
008: move xxx, r2
00C: store 2, (r2)
010: ret

RELOCATION TABLE:
(remember what addresses
  need to be changed)
y: 000
z: 008
```

[foo.s]

Static Linking Example (contd.)

```
int z;
extern int y;

int main() {
        y = 11;
        z = 12;
        bar();
}
```

[bar.c]

```
000: move xxx, r1
004: store 11, (r1)
008: move xxx, r2
00C: store 12, (r2)
010: jsr xxx
014: ret

RELOCATION TABLE:
y: 000
z: 008
bar: 010
```

[bar.s]