Project 3

- Virtual memory trace analysis
- Simulate the VM system over some program
- Two parts:
  - Implement some replacement algorithms
  - Design an experiment to test some aspect of the VM system
- Due: November 23

vmtrace

- Simulate VM given a trace file
    - List of all address refs during execution
- Inputs
    - Trace file
    - Phys. Memory size/page size
    - Replacement algorithm
- Outputs
    - # of references
    - # of faults (incl. # compulsory)
    - Evictions/pageouts

Sample output

```
barb% vmtrace -l 10000 -v random netscape.exe.et.gz
vmtrace: using replacement algorithm 'random'
vmtrace: reading from netscape.exe.et.gz

vmtrace: reached 10000 references
phys_pages,pagesize,input_file,fault_handler,ref_limit
128,1024,netscape.exe.et.gz,random,10000

type,code,load,store
references,6171,1905,1924
miss,62,48,23
compulsory,53,43,21
evictions,27,16,4
pagetouts,9,7,1

barb%
```

Part 1

- Write a series of new replacement algorithms
- Given “random”

```
void fault_random(pte_t *pte, ref_kind_t type)
{
  int page;
  page = random() % opts.phys_pages;
  physmem_evict(page, type);
  physmem_load(page, pte, type);
}
```

Part 1 (cont'd)

- You need to write:
  - FIFO
  - LRU Clock
  - One of your choice
    - True LRU (e.g. via storing full timestamp)
    - Variations on LRU Clock (enhanced second-chance, etc)
    - LFU/MFU
    - Your own!
- You can write more than 3 if your experiment focuses on replacement algorithms.

Part 2

- Have a hypothesis
  - “Algorithm y is better than algorithm x”
  - “Big pages are better”
  - “Prefetching will reduce the number of page faults”
- Explain why you think it will turn out that way
- Two steps
  - Determine baseline behavior
  - New test
    - Change one aspect of the system, observe differences

Part 2 (cont'd)

- What is the ideal page size for this trace under different amounts of main memory?
- Compare performance of various replacement algorithms. How much better/worse is page replacement algorithm X than Y?
    - Compare "real" LRU and LRU clock, FIFO, etc
- How close can we come to LRU without doing any work between page faults?
    - No scanning, constant work per page fault
- How important is recency vs. frequency in predicting page re-use?

Tips

- You control what happens on a page fault
- You control what happens on a memory access
- You can modify formats for PTE, page, etc
- Refresh your scripting skills
- vmtrace is very CPU-intensive
  - forkbomb will be quickly overloaded
  - Find faster machines (such as Linux boxes in the lab)
    - Copy the trace file to local machine
  - Full trace can take hours to execute!