6.508I: Page faults

Plan: Implement VM features using page faults

- Lazy allocation
- Cow vs. fork
- Demand paging

MNAP
Virtual memory benefits

1) Isolation
2) Level of indirection

\[ \text{VA} \rightarrow \text{PA} \]

Static using page faults: we change the mapping
Information needed:

1) the faulting va
   Stval register ← va

2) the type of page fault
   Scause ← w

3) the va of instruction that caused the fault
   sepc ← t + sepc
Allocate: sbrk() → eager allocation

MaxVA → trampoline
  → trapframe
  → heap
  → stack
  → guard page
  → data
  → text

sbrk() ↑

PAGESIZE → P-52

argument 0
... argument N
0
address of argument N
... address of argument 0
address of address of argument 0
argc
0xFFFFFFFF

(EMPTY)
nul-terminated string
argv[argc]
argv[0]
argv argument of main
argc argument of main
return PC for main

applications tend to over ask
Lazy allocation

\[ p \rightarrow s^2 = \text{malloc}(n) \]

Default: va = p \rightarrow s^2

Allocate 1 page
Zero the page
Map the page
Restart instruction
Zero-fill on demand.

Default: Copy + update page

Restart instruction

Exec

BSS

D

initialized

VA

PA
Copy-on-write (cow) fork.  

lab: ref

shell: can't

fork

P

C

exec() echo

default: countdown

copy page, map it
restart instruction

userlet()}
Demand paging

\texttt{execl}():

- load text data segment
- eagerly pgtable

\texttt{pgfault}:

- read block/page from file into mem
- map mem into pgtable
- restart instruction
Demand paging (2)

If out-of-memory:

- evict a page
- use the just-freed page
- restart instruction.

Least-recently-used (LRU)
Memory-mapped files

VMAT = Virtual memory area

\texttt{mmap} \((\text{va, len, prot, flags, \_fd, off})\) 

\texttt{unmap} \((\text{va, len})\)

\texttt{write back dirty block}

\texttt{read\_write}\)

\texttt{fd}\)

\texttt{va}\)

\texttt{sa}\)

\texttt{ld}\)

Latency
Summary

page tables

traps / page fault

→

powerful elegant VM features