(Review Qs delayed - chill out :)

Week 10 Exam - In lab still 45m, still PL

"Final" (Make-ups) - Mon 12-4pm
Wed 10-4pm

Test length 1h 40m
Can retake any or all of the exams

Makeup 1
Makeup 2
Makeup 3

Array Notation	equivalent to...	Pointer Notation
l-values: `a[0]`	(double)	`*a`
l-values: `a[i]`	(double)	`*(a+i)`
`&a[i]`	(double*)	`a+i`

`a[i] = v`

`*(a+i) = v`

`double* a;`

`a += 1`

`a = a+1`

```
void* allocate_at(uint64_t* start, size_t size) {
  size_t current_size = block_size(start[0]);
  if(current_size > size) {
    uint64_t remaining = current_size - size - SLOT_SIZE;
    int next_block_index = (size / SLOT_SIZE) + 1;
    start[next_block_index] = remaining; // even, free
  }
  start[0] = size | 1; // busy
  return &start[1];
}
```

Q1: fill in the blanks in allocate_at w/ pointer notation

Q2: same for free()

```
void* allocate_at(uint64_t* start, size_t size) {

  size_t current_size = block_size(    *start     );

  if(current_size > size) {
    uint64_t remaining = current_size - size - SLOT_SIZE;
    int next_block_index = (size / SLOT_SIZE) + 1;

    *(start + next_block_index) = remaining; // even, free
  }
  start[0] = size | 1;

  return     start + 1     ;
}
```

```
void free(void* ptr) {
  uint64_t* slot_after_header = ptr;
  slot_after_header[-1] = slot_after_header[-1] - 1;
}
```

```
void free(void* ptr) {
  uint64_t* slot_after_header = ptr;
  *(slot_after_header) -= 1;
}
```

```c
#include <unistd.h>
#include <stdio.h>
#include <sys/wait.h>

int main() {
  int value = 17;
  int pid = fork();
  if(pid == 0) {
    printf("In child before assignment: %p => %d\n", &value, value);
    value = 999;
    while(1) {
      printf("In child:\t%p => %d\n", &value, value);
    }
  }
  else {
    printf("In parent before assignment: %p => %d\n", &value, value);
    value = 3333;
    while(1) {
      printf("In parent:\t%p => %d\n", &value, value);
    }
  }
}
```

What prints overall?

How many different addresses print? 1

only 1 address prints

Does this always print 999? yes

Does this always print 3333? yes

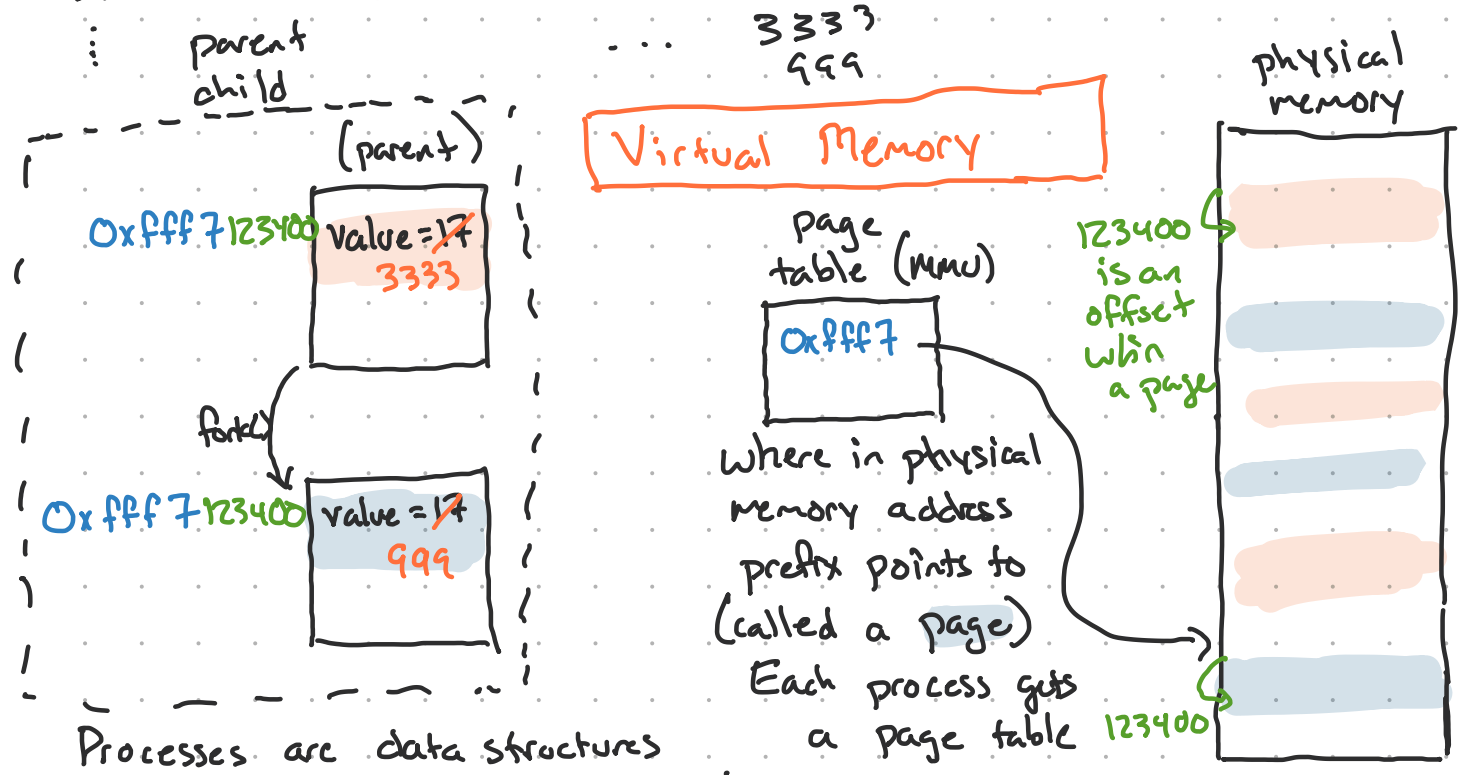
In parent before: 0xfff7123400 => 17

In child before: 0xfff7123400 => 17

In parent: 0xfff7123400 => 3333

In child: 0xfff7123400 => 999

Obs: 1 address is holding 2 diff values?



Processes are data structures in the OS — so are addresses!

Week 10 Exam – just come to lab 45m PL exam

Make ups in finals week. PrairieTest scheduling

- Mon 12-4pm
- Wed 10-4pm

IN CSE LABS

1h40m sessions
Make up any/all exams

Exam 1 Makeup
Exam 2 Makeup
Exam 3 Makeup

Array Notation equivalent to... Pointer Notation

l-value [a[0]] [*a] l-value

a[i] *(a + i)

&a[i] a + i

a[i] = v *(a + i) = v

```
void* allocate_at(uint64_t* start, size_t size) {
  size_t current_size = block_size(start[0]);
  if(current_size > size) {
    uint64_t remaining = current_size - size - SLOT_SIZE;
    int next_block_index = (size / SLOT_SIZE) + 1;
    start[next_block_index] = remaining; // even, free
  }
  start[0] = size | 1; // busy
  return &start[1];
}
```

RQ1: fill in the blanks in allocate_at with pointer notation

RQ2: save for free()

RQ3: draw cool picture ü

```
void* allocate_at(uint64_t* start, size_t size) {
  size_t current_size = block_size( * start );

  if(current_size > size) {
    uint64_t remaining = current_size - size - SLOT_SIZE;
    int next_block_index = (size / SLOT_SIZE) + 1;

*(start + next_block_index) = remaining; // even, free
  }
  start[0] = size | 1;

  return start + 1 ; //pointer arithmetic
}
```

(add 8 bytes to start to get the result) 1 * sizeof(uint64_t)

```
void free(void* ptr) {
  uint64_t* slot_after_header = ptr;
  slot_after_header[-1] = slot_after_header[-1] - 1;
}
```

```
void free(void* ptr) {
  uint64_t* slot_after_header = ptr;
*(slot_after_header - 1) += -1;
}
```

```c
#include <unistd.h>
#include <stdio.h>
#include <sys/wait.h>

int main() {
  int value = 17;
  int pid = fork();
  if(pid == 0) {
    printf("In child before assignment: %p => %d\n", &value, value);
    value = 999;
    while(1) {
      printf("In child:\t%p => %d\n", &value, value);
    }
  }
  else {
    printf("In parent before assignment: %p => %d\n", &value, value);
    value = 3333;
    while(1) {
      printf("In parent:\t%p => %d\n", &value, value);
    }
  }
}
```

What prints?

How many different addresses print?

always print 999?

always print 3333?

In parent before: 0x7fff123400 => 17
In child before: 0x7fff123400 => 17
In parent: 0x7fff123400 => 3333
In child: 0x7fff123400 => 999

Seems like the same address has 2 different values at the same time!