4/4 Questions Answered

Saved at 6:32 PM

Check-in Quiz06, pthreads

Q1 Threads vs Processes

2 Points

(yes, you have done this question before. I'm asking you to do it again as a refresher)

Q1.1

1 Point

Which of these are unique to every thread?

- □ Data in Files on Disk
- □ File Descriptors
- □ Global Variables
- □ Dynamic Storage (The Heap)
- □ Local Variables (The Stack)
- □ CPU Registers

Explanation

Correct! Each thread has its own stack and registers, but share an address space and other attributes that are specific to a process. Note that while each thread has its own stack, a thread could still be given a pointer to some value in another threads stack if we wanted.

Save Answer Last saved on **Dec 05 at 6:32 PM**

Q1.2
1 Point

Which of these are unique to every process?

□ Data in Files on Disk

□ File Descriptors

□ Global Variables

□ Dynamic Storage (The Heap)

□ Local Variables (The Stack)

□ CPU Registers

Explanation

Correct! Each process has its own address space (and the thread(s) within it have their own registers). Each process also has its own file descriptor table, but the actual contents of data in the files are accessible by any process with appropriate permissions

Save Answer Last saved on **Dec 05 at 6:32 PM**

Q2 Shared Memory

4 Points

Consider the code below:

```
// global var
static int counter = 0;

void* thread_fn(void* arg) {
  int amount = *(int*) arg;
  counter += amount;
  free(arg);
  return NULL;
}

int main() {
  pthread_t thd1, thd2;

  int* arg1 = malloc(sizeof(int));
  int* arg2 = malloc(sizeof(int));
  *arg1 = 3;
  *arg2 = 8;

  pthread_create(&thd1, NULL, thread_fn, arg1);
  pthread_create(&thd2, NULL, thread_fn, arg2);

  pthread_join(thd1, NULL);
  pthread_join(thd2, NULL);

  printf("%d\n", counter);

  return EXIT_SUCCESS;
}
```

Select all outputs that are possible. Assume that malloc, pthread_create, and pthread_join all do not error.

□ 0

□ 1

□ 2

□ 6

□ 11

□ 22

□ Compilation Failed

□ Segmentation Fault

Explanation

Correct! Each thread is created and will run, each adding 3 and 8 to the global counter, thus it is possible to get 11. It is also possible to get just 3 or 8, but that will be covered in lecture.

Save Answer Last saved on **Dec 05 at 6:32 PM**

Q3 Thread Ordering

4 Points

Consider the following code that compiles and runs:

```
void* thread_code(void* arg) {
  char* message = (char*) arg;
  printf("%s", message);
  return NULL;
}


int main() {
  pthread_t thd1, thd2;

  pthread_create(&thd1, NULL, thread_code, "A");
  pthread_create(&thd2, NULL, thread_code, "O");

  pthread_join(thd1, NULL);

  return EXIT_SUCCESS;
}
```

Assuming that pthread_create and pthread_join do not fail, what are the possible outputs of this program?

Please answer as a comma separated list with no spaces (sorted alphbetically)
If you think a segmentation fault is possible, write SEGV
for example: `H,HI,I,SEGV`

A,AO,OA

Explanation

Correct! Each thread will be created, but not each thread is joined by main(). This means that the first thread will definitely run, but the second thread may not get a chance to run and print before main returns, which would exit the process. If the second thread is able to run, then we don't know if it will run before the first thread or after, so we can get AO or OA if both threads run.

Save Answer Last saved on **Dec 05 at 6:32 PM**

Save All Answers

Submit & View Submission ›