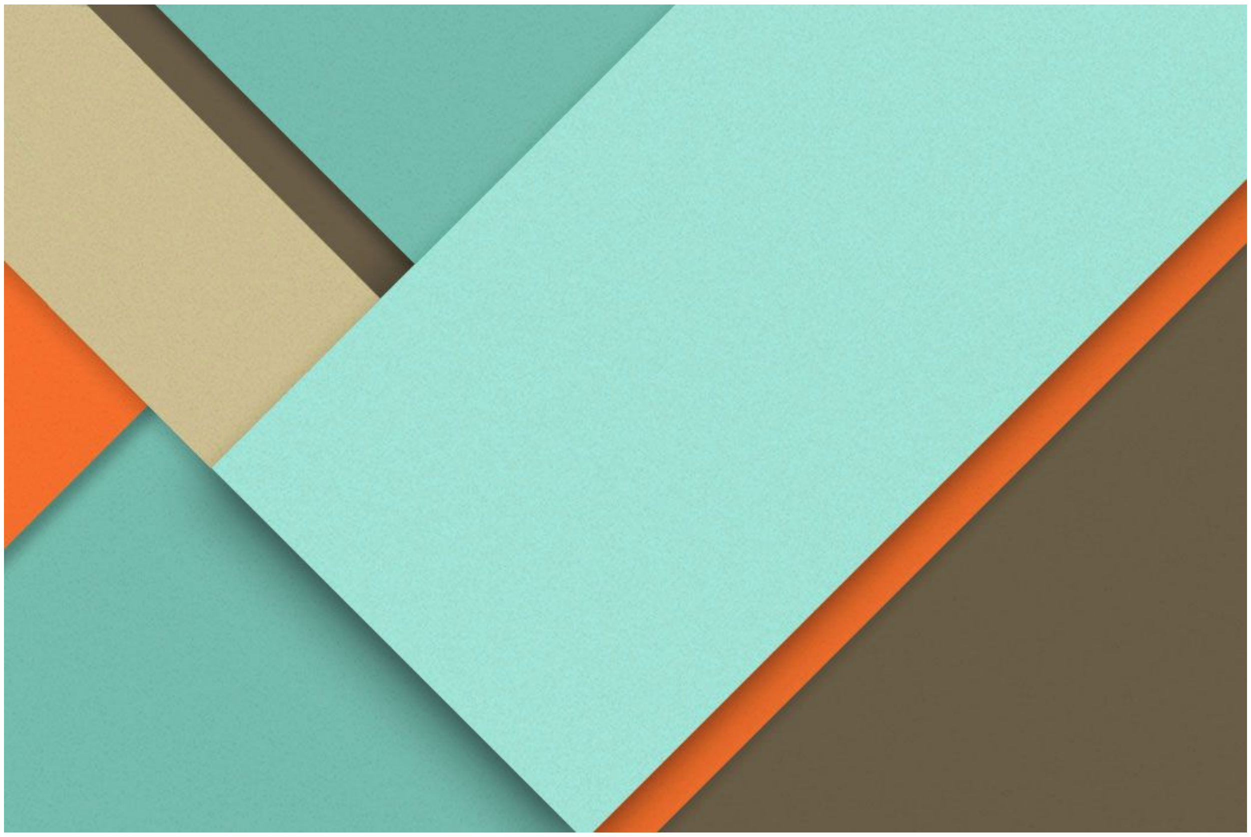

Λειτουργικά Συστήματα Εργασία 2

Ονοματεπώνυμο: Μπαρμπάρ Ηλίας-Ελιάς
AM: 1115201200118
sdi: sdi1200118

Files

mainleit.c/:

We get our cmdl argument, then set our semaphores and shared memory. Then each process gets it's own code segment. Our processes: Constructor2-3, Inspector2-3 have been moved in order to have one of each process types in the forefront: Constructor1, PaintShop, Inspector1, Assembler. All processes begin by attaching the shared memory segments they get to use.

Processes:

- Constructors: Run a loop for all the parts that they are going to make. First we create the part (nanosleep), when it's created we get a timestamp and we set our temp component to use it when we write our the shared memory. Then we semdown the paintshop, when we get access to the paint shop we write on the shared memory segment and signal the PaintShop to start reading, in a bounded buffer fashion. The only difference in the constructor processes is their comp.type assignment.

- PaintShop: Run a loop for all the parts expected (productComponents*productSum). Read the shared memory which contains a part sent from a constructor, semup the semaphore and proceed to start painting that part (nanosleep). When the painting is done, request the Assemblers. Once we get access, write on the shared memory and signal the corresponding inspector to read the shared memory segment.

- Inspectors: Do the same as PaintShop. Run a loop for all parts to be inspected that reads shared memory, semup the semaphore, inspects, writes on shared memory.

- Assembler: Define our lists, one for each part type and one for our products. Read the shared memory segment and add that part to the correct list. If all three lists have a part we create a product. We remove the nodes from the lists (FIFO) while keeping info necessary for our upcoming calculations. Keep all waitForPs from the parts and form the newID our product is going to have then start making the product (nanosleep), get a timestamp when it's made and calculate the time it took from the earliest part to the product being made and then add the product to the list. When the assembler is done with the creation of all the products. He prints the average time for a part to get into the paint shop and and the average time for a product to be made.

functions.c/.h:

We can alter the global int speedup, to change all values that affect nanosleep. (Time it takes for each constructor to create, time it takes for paintshop to paint, time it takes for each inspector to inspect and time it takes to assemble a product.)

Functions:

- findMinTs: Finds the time difference of the product being made and the earliest part made from the parts that helped make this product. First it finds the earliest timestamp from the three provided (from the parts), and then proceeds to calculate the time difference of the earliest timestamp with the products timestamp and then returns the time in ms.

- getTimeDifferenceGivenStructTimespec: Uses the same method as findMinTs with two timestamps instead of three.

- formID: We form an id by getting each integer one by one and placing it in the correct spot in our char array (type ‘c’ id, followed by, type ‘b’ id, followed by type ‘a’ id)

- addID: Returns a random 4 digit number.

- getPsTime: Returns nanoseconds to the PaintShop depending on the part type.

- getInTime: Returns nanoseconds to each Inspector depending on the part type.

- printCmdlErr: Error message for cmdl arguments.

sem.c/.h:

Contains 11 semaphores. 2 for a bounded buffer implementation between Constructors and the PaintShop. 6 for a bounded buffer imp. Between PaintShop and EACH Inspector. 2 for a bounded buffer imp between ALL Inspectors and the Assembler. 3 for mutexes that are used everytime a process writes or reads on a shared memory segment.

Functions: Typical semaphore functions found in our lab codes. Our semaphores come in a set void getSemaphoreSet() and they are all initialized with void initializeSemaphoreSet().

shm.c/.h:

Containsthe struct used for the shared memory segments (Component) which consists of an id, a type, a state, a timestamp which is two double numbers(timestampsec, timestampnsec) and the time waited after creation to access the PaintShop (waitForPs).

Functions:

- setComponent: Gets a component * pointer and all ingredients necessary (except waitForPs) and assign them.
- The rest of the functions are typical shared memory functions, int getSharedMemory(), void *attachSharedMemory(int id), void deleteSharedMemory() (deletes in bulk).

myLists.c/.h:

Contains two very simplistic list creators. One for our part lists and one for our product list. No special function other than basic list function. Implements FIFO.

PartList (myList) Functions: createNode, addNode, removeNode, printList

ProductList (myPrList) Functions: printPrProductMean: Prints the average time for a product to be made, createPrNode, addPrNode, removePrNode, printPrList, deletePrList.

Compilation

We compile using a makefile which has the following form:

```
leit:
	gcc mainleit.c functions.c myLists.c shm.c sem.c -o leit


.PHONY: leit
```

Results

For:

TimeToConstructA: 3ms	TimeToConstructB: 7ms	TimeToConstructC: 5ms
TimeTopaintA: 12ms	TimeTopaintB: 8ms	TimeTopaintC: 16ms
TimeToInspectA: 20ms	TimeToInspectB: 5ms	TimeToInspectC: 13ms

TimeToAssemble: 250 ms

Y = 10:

AverageWaitForPaintShop:	29.7ms
Average Time Needed Per Product:	109.0ms

Y = 100:

AverageWaitForPaintShop:	32.3ms
Average Time Needed Per Product:	142.2ms

Y = 1000:

AverageWaitForPaintShop:	32.8ms
Average Time Needed Per Product:	143.7ms

Y = 2000:

AverageWaitForPaintShop:	32.7ms
Average Time Needed Per Product:	143.4ms

TimeToConstructA: 30ms | TimeToConstructB: 70ms | TimeToConstructC: 50ms
TimeTopaintA: 120ms | TimeTopaintB: 80ms | TimeTopaintC: 160ms
TimeToInspectA: 200ms | TimeToInspectB: 50ms | TimeToInspectC: 130ms
TimeToAssemble: 2500 ms

Y = 10:
AverageWaitForPaintShop: 275.5ms
Average Time Needed Per Product: 1264.0ms

Y = 100:
AverageWaitForPaintShop: 308.1ms
Average Time Needed Per Product: 1348.6ms

Y = 1000:
AverageWaitForPaintShop: 312.1ms
Average Time Needed Per Product: 1365.1ms

Y = 2000:
AverageWaitForPaintShop: 359.3ms
Average Time Needed Per Product: 7847.5ms

TimeToConstructA: 300ms | TimeToConstructB: 700ms | TimeToConstructC: 500ms
TimeTopaintA: 1200ms | TimeTopaintB: 800ms | TimeTopaintC: 1600ms
TimeToInspectA: 2000ms | TimeToInspectB: 500ms | TimeToInspectC: 1300ms
TimeToAssemble: 25000 ms

Y = 5:
AverageWaitForPaintShop: 99.9ms
Average Time Needed Per Product: 2641.3ms

Y = 10:
AverageWaitForPaintShop: 106.6ms
Average Time Needed Per Product: 3652.9ms

Y = 50:
AverageWaitForPaintShop: 126.8ms
Average Time Needed Per Product: 111662.3ms

Y = 100:
AverageWaitForPaintShop: 128.6ms
Average Time Needed Per Product: 21679.9ms