BUS Exercise 7
Group 23

Thilo Metzlaff	Mats Frenk	Emma van Emelen
406247	393702	406008

July 1, 2020

Contents

Introduction

Do i welcome you again or do we know each other well enough by this point? Welp, whatever it is, today's exercise is about fitting big things into small spaces and vice versa. A topic we all understand. After all, we have access to the internet. As long as you don't try to square a circle, try to make a sphere fall over or solve $P = NP$, all is well. This sentence is a Lie.

THE FORMAT

- Every file will be named similar to the sections in here, so `2.1-stack_exercise.c` is Exercise 2, section 1.
- Every Solution **WILL** be in this pdf, but not necessarily anything predefined by the exercise.
- AnysegmentxtbfWARNING: Humor may or may not be used. If you are allergic to humor, that sounds like a personal problem.
- **WARNING:** Backing up your data is important. Although linux doesn't have the necessary shame to remove itself, unlike windows[1], please do back up your data. And try to keep track of your periods... they seem to be notoriously hard to find

[1] Happened to me... too often

1 Fitting things into spaces

1.1 Give it to me

The order of incoming requests, as i understood it, is:

$req_1 \rightarrow$ 1024 Byte

$req_2 \rightarrow$ 256 Byte

$req_3 \rightarrow$ 192 Byte

$req_4 \rightarrow$ 320 Byte

The following will be using this sequence. Now let S_n be the n-th useable memory segment, we get the following results, showing how much memory is free in each segment.

1.1.1 First Fit

	S_1	S_2	S_3
start	1024 Byte	512 Byte	256 Byte
req_1	0 Byte	512 Byte	256 Byte
req_2	0 Byte	256 Byte	256 Byte
req_3	0 Byte	256 Byte	256 Byte
req_4	0 Byte	64 Byte	256 Byte
	Cannot fit into S_2		

1.1.2 Rotating First Fit

I am unsure wether a Segment that is full would be skipped, but it makes sense that it would, so i assumed that it does so.

	S_1	S_2	S_3
start	1024 Byte	512 Byte	256 Byte
req_1	0 Byte	512 Byte	256 Byte
req_2	0 Byte	256 Byte	256 Byte
req_3	0 Byte	256 Byte	64 Byte
req_4	0 Byte	256 Byte	64 Byte
	Cannot fit into S_2		

1.1.3 Expectation - it all fits

This is so far the only algorithm that can allocate these requests properly.

	S_1	S_2	S_3
start	1024 Byte	512 Byte	256 Byte
req_1	0 Byte	512 Byte	256 Byte
req_2	0 Byte	512 Byte	0 Byte
req_3	0 Byte	320 Byte	0 Byte
req_4	0 Byte	0 Byte	0 Byte

1.1.4 Reality - this is bigger than expected

	S_1	S_2	S_3
start	1024 Byte	512 Byte	256 Byte
req_1	0 Byte	512 Byte	256 Byte
req_2	0 Byte	256 Byte	256 Byte
req_3	0 Byte	64 Byte	256 Byte
req_4	0 Byte	256 Byte	64 Byte
	Cannot fit anywhere		

1.2 It's not you, it's me

1.2.1 First Fit

	S_1	S_2	S_3
start	512 Byte	1024 Byte	256 Byte
req_3	320 Byte	1024 Byte	256 Byte
req_4	192 Byte	1024 Byte	256 Byte
req_1	0 Byte	0 Byte	256 Byte
req_2	0 Byte	0 Byte	0 Byte

1.2.2 She loves me, she loves me not

	S_1	S_2	S_3
start	512 Byte	1024 Byte	256 Byte
req_3	320 Byte	1024 Byte	256 Byte
req_1	320 Byte	0 Byte	256 Byte
req_2	320 Byte	0 Byte	0 Byte
req_4	0 Byte	0 Byte	0 Byte

1.2.3 makin it good

No change necessary, this algorithm is already the only one that can allocate the original configuration. I would argue that this can actually allocate most configurations correctly but may sometimes be less efficient, since we have to find the best segment. This would make first-fit most efficient in execution time, since it just allocates wherever it first finds the space, but only useful when we basically have large memory segments, since it is pretty dumb, like my personal memory management, which is basically a garbage dump, so at least i can say i'm sorta similar to a computer.

1.2.4 The Last of us

	S_1	S_2	S_3
start	1024 Byte	256 Byte	512 Byte
req_1	0 Byte	256 Byte	512 Byte
req_3	0 Byte	256 Byte	320 Byte
req_2	0 Byte	256 Byte	0 Byte
req_4	0 Byte	0 Byte	0 Byte

2 Addresses

2.1 Memory block size

To calculate the total size of available memory, we can just add the size of each segment like this:

$$\text{let: } S = \{999, 45, 410, 175, 191, 53, 65\} \tag{1}$$

$$\text{then: } S_{total} = \sum_{n=1}^{|S|} x_i \tag{2}$$

$$\Leftrightarrow S_{total} = 1938 \tag{3}$$

Therefore the size of all