COMPSCI 340 & SOFTENG 370 2017
Operating Systems

Assignment 3 - Memory

Worth 6%

due date 11:59pm Friday the 20th of October

Use any word processor or tools you like for this assignment but you must convert your submission into pdf so that the markers can read it.

Introduction

Most of this assignment is to do with memory, and starts by showing the connection between our executable files and the memory they occupy when they are run.

You will need to use a few of the standard Unix tools to extract this information and also read some explanations of *elf* files in the following web links:

https://en.wikipedia.org/wiki/Executable_and_Linkable_Format

https://blogs.oracle.com/ali/inside-elf-symbol-tables

https://www.cs.stevens.edu/%7Ejschauma/631/elf.html

http://www.skyfree.org/linux/references/ELF_Format.pdf in particular the Program Loading section.

Further information can be gathered from the man pages on `readelf`, and `nm`.

Also read the `/proc/[pid]/maps` section in man `proc`.

Part 1

In this Part there are instructions and questions. Follow the instructions and answer the questions in your assignment document. The answers do not need to be long.

1.1. [2 marks]

Describe the main differences between how program headers and section headers in *elf* files are used.

1.2. [1 mark]

Download the file `hello.c` from the A3 files section of Canvas. Modify `hello.c` so that it starts by saying "`Hello yourUPI`", where `yourUPI` is your login name. Compile it and run it on Linux:

```
gcc hello.c -o hello
./hello
```

Include the output you get in your answer document.

1.3. [2 marks]

Make your terminal window wide then run the command:

```
readelf -aW hello
```

There is a lot of information here. Look through it to get an idea of the different areas. We are not going to examine all of this, but it is good to see what is actually stored in a compiled file. Do NOT include the output from this command in your answer document.

Run the command:

```
readelf -h hello
```

Do include this output in your answer document.

How does the operating system know that this file can be loaded into memory and run?

What does the output tell you about the architecture of the machine this program is intended for?

1.4. [4 marks]

Run the command:

```
nm hello
```

Include the output in your answer document.

For each symbol in the output which also appears in the source code of `hello.c`, explain what type of symbol it is and give its value/address. This includes all symbols which are defined in the source code and those it references in other libraries.

What library or libraries are the undefined symbols in?

Why are there no symbols for any of the local variables?

1.5. [4 marks]

There are 4 special sections of memory associated with C programs: .text, .rodata, .data, .bss. Clearly describe each one.

1.6. [1 mark]

Run the command:

```
readelf -lW hello
```

Include the output you get in your answer document.

What is always true with regards to the virtual addresses of the segments and the offsets in the file?

1.7. [2 marks]

Compare the output in 1.6 with the output from the `hello` program. Produce a small table which shows the relationship between the LOAD segments from the `hello` file and the corresponding memory map output from the `hello` program. The LOAD segments are segments 2 and 3. The table should look like this:

seg	type	virtual address	size	access (flg)	map area (range of addresses)	file offset	mem access
2	LOAD						
3	LOAD						

1.8. [2 marks]

Run the command:

```
readelf -SW hello
```

Using this output and your output from the `hello` program what are the memory access rights for the .text, .rodata, .data, and .bss sections?

Part 2

2.1. [3 marks]

Over the years architectures have changed from 8 to 16 to 32 to 64-bit machines. At the same time the address spaces have moved from 16 (there were never any widely used 8-bit address spaces) to

32 to 64-bit addresses. Give an example of a CPU which provides 64-bit addresses but is backwardly compatible with code that produces 32-bit code.

What consequences does moving from a 32-bit environment to a 64-bit environment have on programs, including the operating system?

2.2. [4 marks]

We can keep track of free memory in a similar way to free blocks on disk. We could use a bitmap where 1 bit represents a free frame of memory or we could maintain a linked list of free frames. The linked list is maintained as a list of nodes where each node has a frame number, and a pointer to the next node. Assume these pointers and numbers are 32 bits each.

Calculate the space requirements using each of those methods for the following system, give the answers in bytes:

16 GB (2^{34} bytes) of RAM in 8 KB (2^{13} bytes) frames. Assume that memory is currently being used in alternating chunks of size 1 MB (2^{20} bytes). i.e. The first MB of RAM is used, the second is free, the third is used, etc.

How much of that space (for both approaches) would normally be in kernel memory? Why?

How would an extents version compare to the two approaches in the question above?

2.3. [2 marks]

We have a system with 32-bit virtual addresses and 1024 processes. Assuming all pages may need to be stored in virtual memory without any sharing, what is the maximum amount of space we should allocate for swap space? How realistic is this amount?

2.4. [1 mark]

You are given the following data about a virtual memory system:

- the TLB can hold 1024 entries and can be accessed in 1 clock cycle, 1 nanosecond.
- memory can be accessed in 50 clock cycles or 50 nanoseconds.
- a page table entry can be found in 100 clock cycles or 100 nanoseconds.
- the average page replacement time is 5 milliseconds.

If page references are handled by the TLB 99% of the time, and only 0.01% lead to a page fault, what is the effective access time?

2.5. [2 marks]

A computer has 64-bit virtual addresses and 4MB (2^{22} byte) pages. For a particular process the program and data together fit in the lowest page. The stack fits in the highest page. Assume no other pages are necessary. How many levels would be required to be allocated for a multi-level page table to cover the full address space, with each page table entry in all levels occupying 2^6 bytes? For our process what is the smallest number of pages that need to be allocated for its page table information, count the page table pages and all necessary indirect page tables. Show your working and give an explanation.

2.6. [2 marks]

The number of instructions executed between page faults is directly proportional to the number of page frames allocated to a program. Double the amount of memory allocated to a process and double the length of time between page faults. Suppose that a normal instruction takes 1 nanosecond, but if a page fault occurs, it takes 2,000,000 nanoseconds (i.e. 2msec) to handle the

fault. If a program takes 60 sec to run, during which time it gets 20,000 page faults, how long would it take to run if twice as much memory was available? Show your working.

2.7. [4 marks]

Given a machine with 5 frames show the contents of each frame using the following replacement algorithms on the reference string. **Also say how many page faults would occur for each algorithm.** Pages are brought in on demand and the initial load of a page counts as a page fault.

a) FIFO - First In First Out

b) LRU - Least Recently Used

c) LFU - Least Frequently Used (if there are multiple pages with the same lowest frequency choose in a FIFO manner)

d) Optimal (if there are multiple pages which are not used again choose in a FIFO manner)

	1	2	3	4	3	5	6	7	6	5	4	3	4	7	6	1	5	4	1	2
0	1																			
0		2																		
0			3																	
0				4																
0						5														

Lay out your answer for each algorithm like this. A zero means the frame is free, an empty cell means the content of the frame is the same as in the previous step.

Submitting the assignment

Make sure your name and upi is included in the file you submit.

Submit your answers to the questions as a single pdf file, called `a3.pdf`. Please do NOT submit a Word, Excel or other proprietary file format.

Any work you submit must be your work and your work alone – see the Departmental and University policies on academic integrity.