2020/2021: Sistemas Operativos (*Operating Systems*) - 2º MIEIC

22.mar.2021

Exercise Sheet 4

Memory theory and practice

1. Consider a system in which the memory, on a given occasion, presents the following empty spaces (holes), in ascending order of position: 10KiB, 4KiB, 20KiB, 18KiB, 7KiB, 9KiB, 12KiB and 15KiB.
    a. Which of those spaces are chosen to satisfy the following successive requests for memory: 12KiB, 10KiB and 9KiB, if the allocation algorithms used are:
        i. "first fit"
        ii. "next fit" (next first fit after "first fit"!)
        iii. "best fit"
        iv. "worst fit"?
    b. For this example, order the allocation methods regarding the waste of memory space they cause (external fragmentation).
2. State an advantage and a disadvantage of using paging over variable memory partitions in a virtual memory system.
3. In the lecture's slides (page 14) it was asked why not have a Table mapping page 0 of each process to the physical address of the corresponding page frame instead of maintaining a Page Table mapping all pages of each process to the corresponding page frames. The mapping table would be smaller, so the translation performance of this "solution" would be better, would it not be?... So, why is it not used?
4. Write a program, called `mt` (*memory test*) to test the limits of memory (physical and virtual) of a computer system. To do this, it is suggested that the program keeps calling `malloc()` every second, asking for 50MB chunks, until the system denies a request.
   While `mt` runs, observe the overall memory of the system using the `vmstat` utility, operating in a periodic output mode, (e.g. 1 s) or using a graphical tool (e.g. KDE's `kinfocenter`).
   (You could also get information on system's memory through `/proc/meminfo` : you could use a script to periodically read the file and filter the output to your liking.)
    - Now repeat the procedure, but with a difference: as the system hands over a chunk, fill it out with a char (e.g. 'A'). You can use `memset()` for that.
5. Study the information provided by utilities such as `vmstat` and `time` (the executable in disk, not shell's builtin `time` command) that monitor memory activity and disk access. Try the utilities with different programs, running each program several times in a row.
    - For instance, `time` outputs to *standard error* interesting information on memory activity concerning the program presented as argument, after it run. See what can be learned with `time`'s compound option:
        - `-f "\nsize: %D ; %K ; %X ; %p (Kilobytes)\npaging: %Z (bytes) ; %F ; %R ; %M ; %t (Kilobytes)\nswap: %W\n"`
6. Discuss the possibility of programmatically evaluating the size of the pages used by the paging method of a given system and write a program that will

test your idea. Compare the results you get with those collected directly by calling `sysconf(_SC_PAGESIZE)` or from `time`'s resource specifier `Z`.

7. Try to demonstrate the memory layout of a C program, usually presented on the Net (and on SOPE's lecture on Programming Basics) by experimenting with a simple toy program and using on it some Unix tools, such as `size` (option `-t` is neat).
   - Start by following the article in `www.geeksforgeeks.org/memory-layout-of-c-program/` ;
   - then explore further by extending the program and trying to print the memory addresses of functions and variables that you know will be in the `text`, `stack` and `heap` regions.
8. Study the following program, add the necessary "includes" and try it. Then, spot the bugs (just by looking at the code and with tools such as Infer and Valgrind) and write a new, debugged program.

```
void *garbage_collected_malloc(size_t size) {
  void *p = malloc(size);
  free(p);
  return p;
}

int main(int argc, char **argv) {
  char *buf;
  int i;

if (argc < 2) {
  fprintf (stderr, "\nUsage: %s <string>\n", argv[0]);
  exit (1);
  }
printf("Input string: %s\n", argv[1]);
buf = (char *) garbage_collected_malloc(10);
strncpy(buf, argv[1], 10);
for (i=0; i< 10 ; i++) {
  buf[i] = toupper(buf[i]);
  }
printf("Output string: %s\n", buf);
return 0;
}
```

9. Consider the following "bugged" program (taken from `www.thegeekstuff.com`), that is ineffective for controlling the access to a privileged section of the code by means of knowledge of a password.

```
#include <stdio.h>
#include <string.h>

int main(void) {
     char buff[15];
     int pass = 0;

  printf("\n Enter the password : \n");
  gets(buff);
  if(strcmp(buff, "thegeekstuff")) {
     printf ("\n Wrong Password. \n");
    }
  else {
     printf ("\n Correct Password.\n");
     pass = 1;
    }

  if(pass) {  // do somelhing priviledged stuff
     printf ("\n Root privileges given to the user.\n");
    }
  return 0;
```

```
}
```

   a. Explain how the protection can be defeated by someone with no knowledge of the password.
   b. Correct the bug so that only with knowledge of the password can a user execute the privileged section of the code.
10. Repeat the previous problem, but moving local variable `buff[]` to inside a (new) function. Then try to *smash the stack* (most famous Buffer OverFlow attack) by repeating here your experiences with the previous problem. You should see different system behavior with some string length variation, but all that might depend much on the system you will use.
11. State an advantage and a disadvantage of a virtual memory system with segmentation only over one with pagination only.
    - Will a mixed system (pagination with segmentation), taking advantage of both techniques, be the "ideal" solution?
12. Consider a system that implements paginated memory with (only!) 4 page frames. The table shows the paging information available to the operating system at the time a frame release decision has to be made.

frame	page	loaded (*clock ticks*) ago	referenced (*clock ticks*) ago	R flag	M flag
0	35	126	280	1	0
1	12	230	265	0	1
2	9	140	270	0	0
3	87	110	285	1	1

Which of the pages shown will be replaced next when the algorithm used by the operating system for the replacement is:

   i. "NRU"
   ii. "FIFO"
   iii. "LRU"
   iv. "FIFO, second chance"?
13. ... (additional exercises on Moodle for people with free time)