UNIVERSITEIT TWENTE.

Examination Operating Systems

Bachelor year 2, Computer Science, EWI

Module/course code:	Computer Systems
Date:	6 November 2015
Time:	13:45-15:45 (+25% for students who may use extra time)
Module-coördinator:	Andre Kokkeler
Instructor:	Pieter Hartel
100% of the marks:	60 credits

Type of test:

- Closed book

Allowed aids during the test:

- Nothing

Attachments:

- Various Linux Manual pages

Additional remarks:

- Read these instructions and the questions carefully! If the questions are unclear, you can ask for clarification.
- Please make sure that your name and student number appear on all answer sheets.
- Try to give precise answers using appropriate terminology. For multiple-choice questions there may be more than one correct answer; all of these must be selected for full marks.
- Unreadable or extremely long answers will not be marked. Multiple-choice answers that are ambiguous will not be marked either.
- You are only allowed to use your writing materials and the Linux man pages provided during the exam.
- Feel free to give your answers in English.

Nr: 2.6

Q: Given the following C-program fragment:

```
extern char etext, edata, end;
int a = 0xaaaa, b;

int main(int argc, char * argv[]) {
    int c = 0xcccc;
    int *d_ptr = (int*) malloc(sizeof(int));
    int *e_ptr = (int*) alloca(sizeof(int));
    b = 0xbbbb;
    *d_ptr = 0xdddd;
    *e_ptr = 0xeeee;
    printf("%p:a=%0x\n", &a, a);
    printf("%p:edata\n\n", &edata);
    printf("%p:b=%0x\n", &b, b);
    printf("%p:end\n\n", &end);
    printf("%p:d=%0x\n", d_ptr, *d_ptr);
    printf("%p:brk\n\n", sbrk(0));
    printf("%p:e=%0x\n", e_ptr, *e_ptr);
    printf("%p:argc=%0x\n", &argc, argc);
    printf("%p:c=%0x\n\n", &c, c);
    printf("%p:main\n", &main);
    printf("%p:etext\n\n", &etext);
    return 0;
}
```

(a) Give an example of the output of the program. Here the actual addresses are not important (you may even choose to use 1, 2, 3 etc), but the order of the addresses must be correct.

(b) Annotate each segment of the program's address space with its conventional Unix/Linux name.

(c) If you would run the program several times on a current Linux system with the latest protection, then which addresses would change and which would remain the same? Why?

C: 7 credits

Nr: 3.5

Q: Using the following program, state the PID values printed in lines A, B, C, and D. Assume that the actual PID of the parent process is 2600 and of the child 2603.

```
int main() {
pid_t  pid, pid1;
     pid = fork();
     if (pid < 0) {
       fprintf(stderr, "Fork Failed");
       exit(-1);
     } else if (pid > 0) {
       pid1 = getpid();
       printf("A: %d\n", pid);  /* A */
       printf("B: %d\n", pid1); /* B */
     } else {
       pid1 = getpid();
       printf("C: %d\n", pid);  /* C */
       printf("D: %d\n", pid1); /* D */
     }
     return 0;
}
```

C: 7 credits

Nr: 3.8

Q: Consider the C program fragment below (**Read this very carefully!**):

```
void sigsegv_handler(int sig) {
     printf("SIGSEGV received.\n");
     exit(0);
}

int main(int argc, char * argv[]) {
     int *foo = (int *) & foo;
     signal(SIGSEGV,sigsegv_handler);
     printf("About to segfault:\n");
     *foo=0;
     printf("Why didn't we crash?\n");
     return 1;
}
```

(a) What is the actual output of the program? Why?
(b) Is the exit function ever called? Why?

C: 7 credits

Nr:	4.8

Q: Consider the C and Java program fragments below:

```
#define N 5000
void* tproc(void *arg) {
    printf("Thread %d\n", *((int *)arg));
    return NULL;
}
int main(int argc, char * argv[]) {
    int i;
    int targ[N];
    pthread_t tid[N];
    for(i = 0; i < N; i++) {
        targ[i] = i*i;
        pthread_create(&(tid[i]), NULL, &tproc, &targ[i]);
    }
    for(i = N-1; i >= 0; i--) {
        pthread_join(tid[i], NULL) != 0;
    }
    return 0;
}
```

```
class MyThread extends Thread {
    static final int N = 5000 ;
    int arg;
    public MyThread(int arg) {
        this.arg = arg;
    }
    public void run() {
       System.out.println("Thread " + arg);
    }
    public static void main(String [] args) {
        MyThread[] tid = new MyThread [N] ;
        for(int i = 0; i < N; i++) {
            tid[i] = new MyThread(i*i);
            tid[i].start();
        }
        for(int i = N-1; i >= 0; i--) {
            try { tid[i].join(); }
            catch(InterruptedException e) { }
        }
    }
}
```

(a) What is the main difference between the outputs produced by the two programs? Why?

(b) Which of the two programs is faster? Why?

C:	7 credits							

Nr:	4.9

Q: Consider the C program fragment below:

```
#define N 5
void* tproc(void *arg) {
     printf("Thread %d\n", *((int *)arg));
     return NULL;
}
int main(int argc, char * argv[]) {
     int i;
     int targ[N];
     pthread_t tid[N];
     for(i = 0; i < N; i++) {
          targ[i] = i*i;
          pthread_create(&(tid[i]), NULL, &tproc, &targ[i]);
     }
     for(i = 0; i < N; i++) {
          pthread_join(tid[i], NULL);
     }
     return 0;
}
```

(a) How many threads will be created when this program is executed by the shell? Why?
(b) What is the output of the program? Why?
(c) Is the output always the same? Why?
(d) Explain why the argument of the `printf` statement is not simply: `* arg`.

C:	7 credits						

Nr: 6.8

Q: A semaphore satisfies the following invariants:

$$S \geq 0$$

$$S = S_0 + \#Signals - \#Waits$$

where

S_0 is the initial value of S

#Signals is the number of executed Signal(S) operations

#Waits is the number of completed Wait(S) operations

Given the two concurrent processes below, prove the mutual exclusion property, using the two semaphore invariants. S_0 is initialised to 1 before the processes start.

```
while(true) {
  a1: Non_Critical_Section_1;
  b1: Wait(S);
  c1: Critical_Section_1;
  d1: Signal(S);
}
```

```
while(true) {
  a2: Non_Critical_Section_2;
  b2: Wait(S);
  c2: Critical_Section_2;
  d2: Signal(S);
}
```

C:	7 credits						

Nr: 7.4

Q: Consider the C program fragment below:

```
 7  #define N 2
 8  #define P 3
 9  sem_t Room; /* Initialised to P-1 */
10  sem_t Fork[P]; /* Each semaphore initialised to 1 */
11  void *tphilosopher(void *ptr) {
12      int i, k = *((int *) ptr);
13      for(i = 1; i <= N; i++) {
14          printf("%*cThink %d %d\n", k*4, ' ', k, i);
15          sem_wait(&Room) ;
16          sem_wait(&Fork[k]) ;
17          sem_wait(&Fork[(k+1) % P]) ;
18          printf("%*cEat %d %d\n", k*4, ' ', k, i);
19          sem_post(&Fork[k]) ;
20          sem_post(&Fork[(k+1) % P]) ;
21          sem_post(&Room) ;
22      }
23      pthread_exit(0);
24  }
25
26  int main(int argc, char * argv[]) {
27      int i, targ[P];
28      pthread_t thread[P];
29      sem_init(&Room, 0, P-1);
30      for(i=0;i<P;i++) {
31          sem_init(&Fork[i], 0, 1);
32      }
33      for(i=0;i<P;i++) {
34          targ[i] = i;
35          pthread_create(&thread[i], NULL,
36              &tphilosopher,(void *) &targ[i]);
37      }
38      for(i=0;i<P;i++) {
39          pthread_join(thread[i], NULL);
40      }
41      return 0;
42  }
```

(a) Give an example of the output of the program.

(b) What would happen if the Room semaphore was left out? Why?

(c) Does the order of the wait calls in lines 15-17 matter? Why?

C: 7 credits

Nr:	9.3					
Q:	(a) What are the benefits that virtual memory can offer? (List no less than two benefits) (b) Give a short explanation of demand paging. (c) When a program is started, what will happen if one needed page is not in the physical memory? Given that there is one free frame available in the free-list, what are the procedures taken by operating system to handle this kind of situation?					
C:	7 credits					

Nr:	14.1					
Q:	Which of the following statements on access control matrixes are correct? (a) Rows represent objects (b) Rows represent domains (c) Columns represent objects (d) Columns represent domains (e) Can be expanded to include dynamic protection (f) Cannot be expanded for dynamic protection (g) Access control matrix are not used in modern operating systems (h) Separates the mechanism from the policy					
C:	2 credits					

Nr:	15.6					
Q:	(a) Define confidentiality (b) Define integrity (c) Define availability (d) These three terms together are usually referred to as the .... ?					
C:	2 credits					

C i A 1 p