Vika 6

TÖL401G - Stýrikerfi

1. Identify the basic components of a thread, and contrast threads and processes.

- A thread is a basic unit of CPU utilization; it comprises a thread ID, a program counter, a register set, and a stack. Creation of a thread and context switch is more efficient/faster than that of a process. Threads share the same address space, while processes have their own address space. Threads are used for concurrency, while processes are used for (full) parallelism. Threads are more lightweight than processes.

2. Describe the major benefits and significant challenges of designing multithreaded processes.

- Benefits:
  – **Responsiveness**, a multy threaded process may start one thread for computation, one thread for user interaction, etc
  – **Resource sharing**, memory of process is shared between threads, no system calls required for creating shared memory area or for message passing
  – **Economy**, creating threads is faster than creating processes. Context switch (between threads of same process) is faster for threads than for processes
  – **Scalability**, each thread can be executed by a diffrent processor/core, achieving a speed-up by parallel processing.
- Challenges:
  – **Dividing activities**, which activities can run in parallel
  – **Balance**, overhead of thread handlingcommunicationsyncronisation may outweigh performance gain
  – **Data splitting**, not only a challange how to split activities, but also how to divide data processed by diffrent threads
  – **Data dependency**, if a thread depends on data produced by another thread, syncronisation between threads is needed
  – **Testing and debugging**. inherently more difficult than single-threaded applications.

3. Describe different multithreading models.

- **Many-to-one model:**
  – many user-level threads mapped to one kernel thread.
- **One-to-one model:**
  – one user-level thread mapped to one kernel thread.
- **Many-to-many model:**
  – many user-level threads mapped to many kernel threads.
- **Two-level model:**
  – a combination of the many-to-one and one-to-one models.

4. Design multithreaded applications using the POSIX Pthreads and Java threading APIs.

- create new thread and return its thread Id:

```
pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void
*arg);
```

- Thread hands over control to thread library:

```
int pthread_yield()
// or
int sched_yield()
```

- Thread terminates itself:

```
void pthread_exit(void *retval)
```

- Thread waits for termination of another thread:

```
int pthread_join(pthread_t thread, void **thread_return)
```

- Java thread creation:

```
class MyThread extends Thread {
    public void run() {
        // code to be executed in new thread
    }
}

class MainThread {
    public static void main(String args[]) {
        MyThread t = new MyThread();
        t.start();
    }
}
```

5. Have heard about implicit threading approaches.

- **Implicit threading:**
  – threads are created and managed by compilers and runtime libraries instead of programmer.
- Advantages:
  – programmer does not need to worry about thread creation, management, and synchronization.
  – compiler and runtime library can decide how to map threads to processors.
- Disadvantages:
  – programmer has less control over thread management.
  – compiler and runtime library may not be able to determine how to parallelize the code.

6. Know about issues of using threads.

- Calling process may hace created multible threads
- Design decisions to make (fork() copies all threads or only calling thread)
- Either a thread terminates itself or is terminated by another thread before process terminates