Operating Systems
processes and threads

Gerhard Fohler
rts.eit.uni-kl.de/os

• terms and concepts
• states of processes
• control structures in OS
• processes – threads – OS
Process

- executable program
- used data (variables, etc.)
- context
  - current state of processor
    • PC, registers, ...
  - data for process handling
    • priority, ...

Trace

- characterizes behavior of process
- sequence of instructions which are executed for process
- overlapping of traces of different processes characterizes process behavior
Traces

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a+0</td>
<td>b+0</td>
<td>c+0</td>
</tr>
<tr>
<td>a+1</td>
<td>b+1</td>
<td>c+1</td>
</tr>
<tr>
<td>a+2</td>
<td>b+2</td>
<td>c+2</td>
</tr>
<tr>
<td>a+3</td>
<td></td>
<td>c+3</td>
</tr>
<tr>
<td>a+4</td>
<td></td>
<td>c+4</td>
</tr>
<tr>
<td>a+5</td>
<td></td>
<td>c+5</td>
</tr>
<tr>
<td>a+6</td>
<td></td>
<td>c+6</td>
</tr>
<tr>
<td>a+7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a+8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a+9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

I/O

CPU

simple process modell

enter not running dispatch running exit

pause

queue

enter dispatch exit

pause
OS and Processes

- OS controls execution of processes, execution patterns, resource allocation
- Representation of processes has to support OS
  - State of process
  - Memory used by process
  - Maintenance of processes in (state) queues

Creation of Processes

OS builds data structures and allocates necessary memory

When is a process created?
- Login of interactive user
- OS: performing a service
- Creation by user process
  - Process spawning, parent, child
Termination of processes

process indicates termination
• Logout by user
• service request to OS
• occurrence of fault during execution of process
• halt-instruction

Reasons for Termination

• program done
• exceeded time limit
• insufficient memory
• memory access violation
• unauthorized access
• arithmetic fault
• exceeded waiting time
• I/O fault
• invalid instruction
• privileged instruction
• operator or OS intervention
• termination of/by parent process
Major Process States

• **running**
  – process has CPU and is executing

• **ready**
  – process is ready to execute (but waits for access to CPU)

• **blocked**
  – process waiting for event, e.g., completion of I/O, not ready to run

Major Process States

• **New**
  – OS has created process
    • processnummer (process identifier)
    • tables and entries for process control
  – but process not yet ready to run
    • avoid resource overload due to too many processes
Major Process States

- **Exit**
  - termination
  - process no longer executed
  - process information (tables) maybe used by auxiliary programs, e.g., accounting
  - process information and tables deleted if no longer needed

5 state process model

- **new**
- **ready**
- **running**
- **exit**
- **blocked**

- **Admit**
- **Dispatch**
- **Release**
- **Timeout**
- **Wait**
**Process Switch**

- switching of running process
- possible when OS has CPU:
  - Supervisor Call
    - explicit call by program, e.g., I/O
  - Trap
    - error
  - Interrupt
    - cause outside process, control given to interrupt handler and OS
Swapping

- too many processes in main memory lead to decreased performance
- approach: Swapping = move processes to secondary storage (and move back again to main memory)
- in OS: two new process states and queues
  - ready, suspend
  - blocked, suspend

process states with suspend
Causes of Suspend

- Swapping
- BS moves process to secondary storage, e.g., background process, utility, process with problems
- interactive request, e.g., debugging
- timing: periodic process can be moved out in between activations

- terms and concepts
- states of processes
- control structures in OS
- processes – threads – OS
Control Structures for Process Management in OS

Control structures in OS

OS manages the following tables for processes and resource:

- memory tables
- I/O tables for devices and channels
- file tables
- process tables
Process Image

- user program
- user data
  - modifiable part of user space
    (daten, user stack, modifiable programs)
- system stack
  - parameters and calling address of system calls
- process control block (PCB)
  - execution context
  - process identification, processor state information, process control information

Process Image

- program as seen from OS

  process id.
  processor state information
  process control information
  user stack
  private user address space
  shared address space

  as if program was alone
Process Image

- Process Control Block (PCB)
  - is virtual memory
    - not necessarily continuous
  - Primary Process Table pointer to process identifier (PI)
  - OS needs parts of image in main memory for process management

PCB: Process Identification

- process id.
- processor state information
- process control information
- user stack
- private user address space
- shared address space

- unique process number (proc. identifier)
  - index in primary process table
- user identifier
  - user who owns process
- Parent Process Identifier
  - number of process who generated this process
PCB: Processor State Information

- register contents
- control- and status register
  - program counter
  - program status word (PSW)
    - information on control, modus, status-bits
- Stack Pointers

PCB: Process Control Information

- scheduling and state information
  - current state of process
  - priority and scheduling information
  - event, for which process waits
- pointers to other processes
  - process-queues
  - parent, child, …
PCB: Process Control Information

- Interprocesscommunication (IPC)
  - flags, signale, pointers to messages
- privileges
- memory management
  - pointers to segment or page tables
- resources
  - used: open files, I/O devices
  - already consumed: CPU time, I/O, etc.

**Execution Modes**

- at least two *execution* modes in order to protect datastructures of OS (e.g., PCB)
  - privileged mode (= system mode, kernel mode, supervisor mode, control mode)
  - access to control registers, ...
- User Mode

  - mode switch in routines of OS
  - execution modes supported by mode-bits of CPU
**Mode Switch vs. Process Switch**

- different concepts
- mode switch basis for process switch
- not every mode switch results in process switch

**Mode Switch**
- call, trap, interrupt
- save/restore processor context

**Process Switch**
- update PCB (state)
- PCB → queue
- select new process
- update PCB
- update MM data

**Terms and Concepts**
- states of processes
- control structures in OS
- processes – threads – OS
Processes in Operating System

OS and Processes

different approaches to implement OS
1. strict separation of OS kernel and processes
2. OS executes inside user processes
3. process based OS
Processes and Kernel decoupled

- concept of process only for user programs
- OS operates separated from processes in privileged mode

OS execution in processes

- OS is collection of routines
- almost all OS routines executed in process context after mode switch
OS execution in Processes

- OS code and Data in shared address space abgelegt
- separate kernel stack for kernel mode
- user program as well as OS routines running in process

Process based OS

- OS is collection of system processes
- OS provides only basic services (process switching; memory management, IPC, interrupts und I/O) not separate process
- Process Switching? ⇒ seeThreads
Summary Processes

- process is central concept of OS
- OS creates, manages, and terminates processes
- process going through number of states (ready, running, blocked, suspend, ...)
- data structures for process management
  - process table
  - process image: used memory space, PCB: state, resources, priority, etc.

Summary Processes

- mode switch between user/kernel Mode
  - interrupt, trap or supervisor Call
- approaches to implement OS and processes
  - strict separation
  - OS functions in user processes
  - kernel processes for execution of OS functions
Threads

Motivation Threads

- processes are units for both
  1. resource management
  2. dispatching (starting, preemption, terminating tasks, see later)
- processes are big, overheads
- decouple (1) and (2):
  - \textit{Process (Task)}: unit for resource management
  - \textit{Thread (Lightweight Process)}: unit for dispatching, small overhead
- \textit{Multithreading}: \( n > 1 \) Threads per process
Processes and Threads

- process
  - virtual address space with process image
  - memory protection, files, I/O, resources
- Thread
  - state (running, ready, …)
  - context
  - Stack
  - thread-locale variables

Single-threaded Process Model

Process

- Process Control Block
- User Stack
- User Address Space
- Kernel Stack
Thread vs. process

- thread creation takes less time
- switching between threads less overhead, time than process switch
- terminating threads takes less time
- communication between threads of a process can be done without bothering kernel (careful with synchronization)
Use of von Threads

(small) applications working together/on larger application
• e.g., file server in LAN
  – high number of requests in short time
  – process data/resources the same
  – one thread per request
• e.g., Spreadsheet
  – one thread to display menus and read input
  – one thread for calculations
  – one thread for updates…

States of Threads

• basic states: running, ready, blocked
• Suspend not for single threads – all threads of a process have access to same adress space
• terminating a process terminates all its threads
• who is blocking?
  – single thread?
  – whole process?
User-Level Threads (ULT)

- threads not visible to kernel
- thread management via thread library
- thread switching in user mode (mode switch not needed)
- application specific scheduling

Threads Library

- contains code for
  - creation and termination of threads
  - data and message exchange among threads
  - thread scheduling
  - saving and restoring thread context
- blocking system call blocks all ULTs of a process
- no distribution to several processors
Kernel-Level Threads (KLT)

- thread management by Kernel
- kernel thread API, no Library
- thread switching done by Kernel
- scheduling of threads

Kernel-Level Threads

- thread wise blocking
- kernel - routines multi-threaded
- can schedule several threads of a process (multi processor, multi core)
- thread switching within process via needs 2 Mode Switches
  ⇒ slower than ULTs
  ⇒ e.g. combined ULT/KLT
Summary Threads

- thread: unit for dispatching
- several threads per process possible
- faster creation and switching than with processes
- implementations
  - user-level threads
  - kernel-level threads

Ausblick

- processes, threads
- par, Synchronisation (Mutual Exclusion)
- Deadlock
- Speicherverwaltung (Virtual Memory)
- Scheduling
- Ein- und Ausgabe
- Dateiverwaltung