Interprocess Communication (IPC)

Andre M. Maier, DHBW Ravensburg
dhbw@andre-maier.com

Interprocess Communication (IPC)

- Describes mechanisms which allow processes to intercommunicate and synchronize their actions.
- Applications that use IPC can be categorized as clients and servers.
- Processes within a system may be independent or cooperating
- Processes may be running on one or more computers connected by a network

DHBW
Duale Hochschule
Baden-Württemberg

Andre M. Maier
Dept. of Computer Science

IPC Approaches

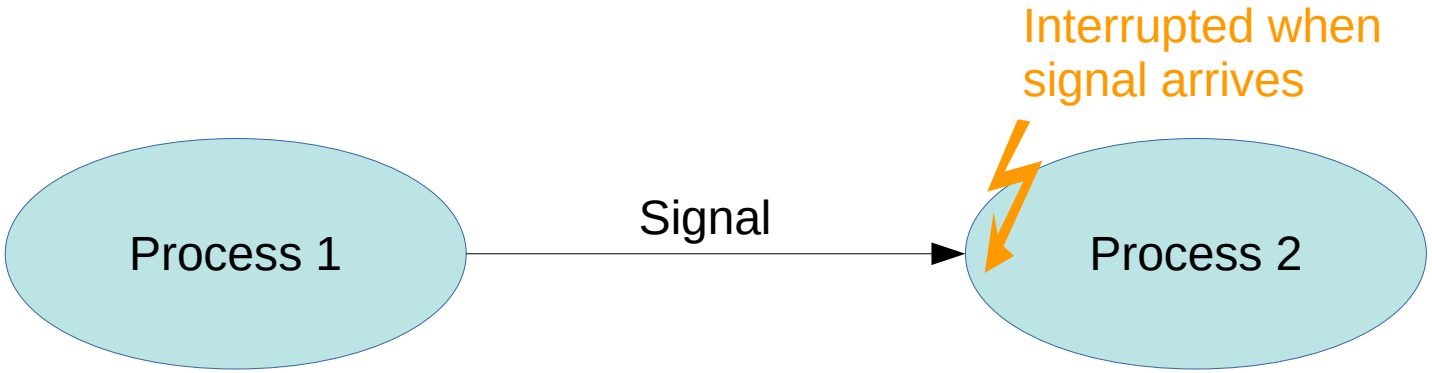
- File
- Signal or Asynchronous System Trap (AST)
- Socket
- Unix Domain Socket
- Message Queue
- Pipe
- Named Pipe
- Shared Memory
- Message passing
- Memory-Mapped File

DHBW
Duale Hochschule
Baden-Württemberg

Andre M. Maier
Dept. of Computer Science

Signals (1)

- A signal is an asynchronous notification a process can sent to another one.
- The execution of the target process is interrupted upon signal delivery.
- There are different signal types, each of which is identified by a signal number and a mnemonic name.
- Signals do not carry any arguments.



DHBW Duale Hochschule Baden-Württemberg
Andre M. Maier
Dept. of Computer Science

Signals (2)

- Signals in Linux

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

DHBW
Duale Hochschule
Baden-Württemberg

Andre M. Maier
Dept. of Computer Science

Signals (3)

- Default signal handler
  - either ignores the signal or causes the target process to die
- Signal handler defined by process
  - provides customized signal handler code that is executed upon reception of the signal
  - OS does not allow programmers to catch and handle signals essential to the integrity of the system, such as SIGKILL

Andre M. Maier
Dept. of Computer Science

Signals (4)

- Sending signals
  - `kill` or `killall` shell commands
  - Bash shortcuts, such as CTRL-C
  - `int kill(pid_t pid, int sig)` in C
- Handling signals
  - `sighandler_t signal(int signum, sighandler_t handler)` in C

DHBW Duale Hochschule Baden-Württemberg
Andre M. Maier
Dept. of Computer Science

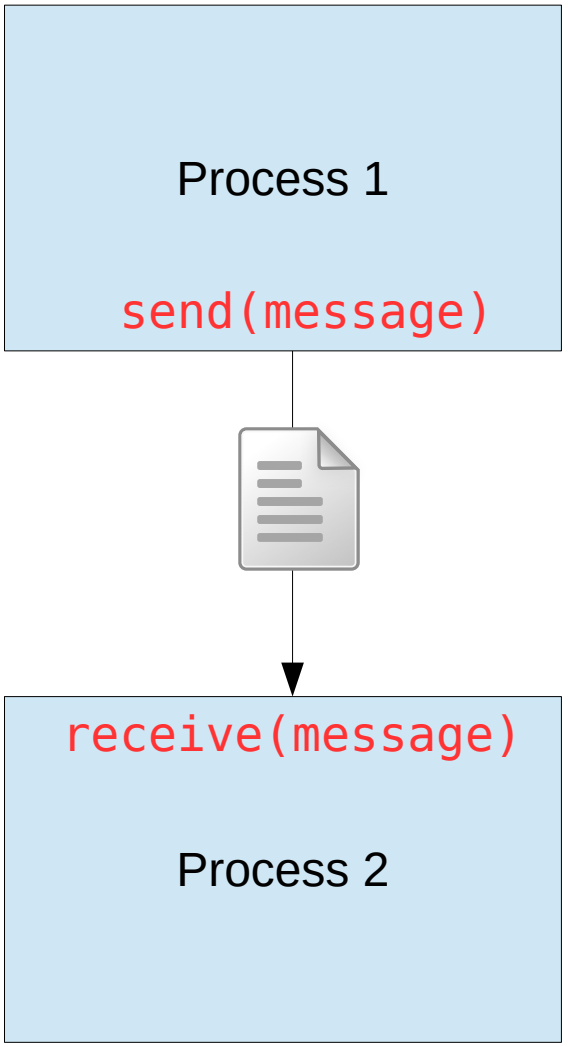
Named Pipes

- Named pipes, a.k.a. Unix FIFOs, are pipes that are mapped into the file system.
- Named pipes can be created by using a
  - Shell command
    `mkfifo [OPTION]... NAME...`
  - C library function
    `int mkfifo(const char *pathname, mode_t mode)`
- Named pipes look like normal files, but they are not.
  `prw-r--r-- 1 cory cory 0 Jan  6 15:25 namedpipe`

DHBW Duale Hochschule Baden-Württemberg
Andre M. Maier
Dept. of Computer Science

Message Passing (1)

- Allows processes to communicate with each other by using two operations:
  - send(message)
  - receive(message)
- Communication can either be *direct* or *indirect*.



DHBW
Duale Hochschule
Baden-Württemberg
Andre M. Maier
Dept. of Computer Science

Message Passing (2)

- Direct Message Passing
  - Sender “knows” the receiver’s identity and sends the message directly to the receiving process.
- Indirect Message Passing
  - The message is sent to a mailbox (or port) where it is stored until the receiving process retrieves it.
  - For indirect message passing, the message passing system must have some buffering mechanism to store the messages.

Andre M. Maier
Dept. of Computer Science

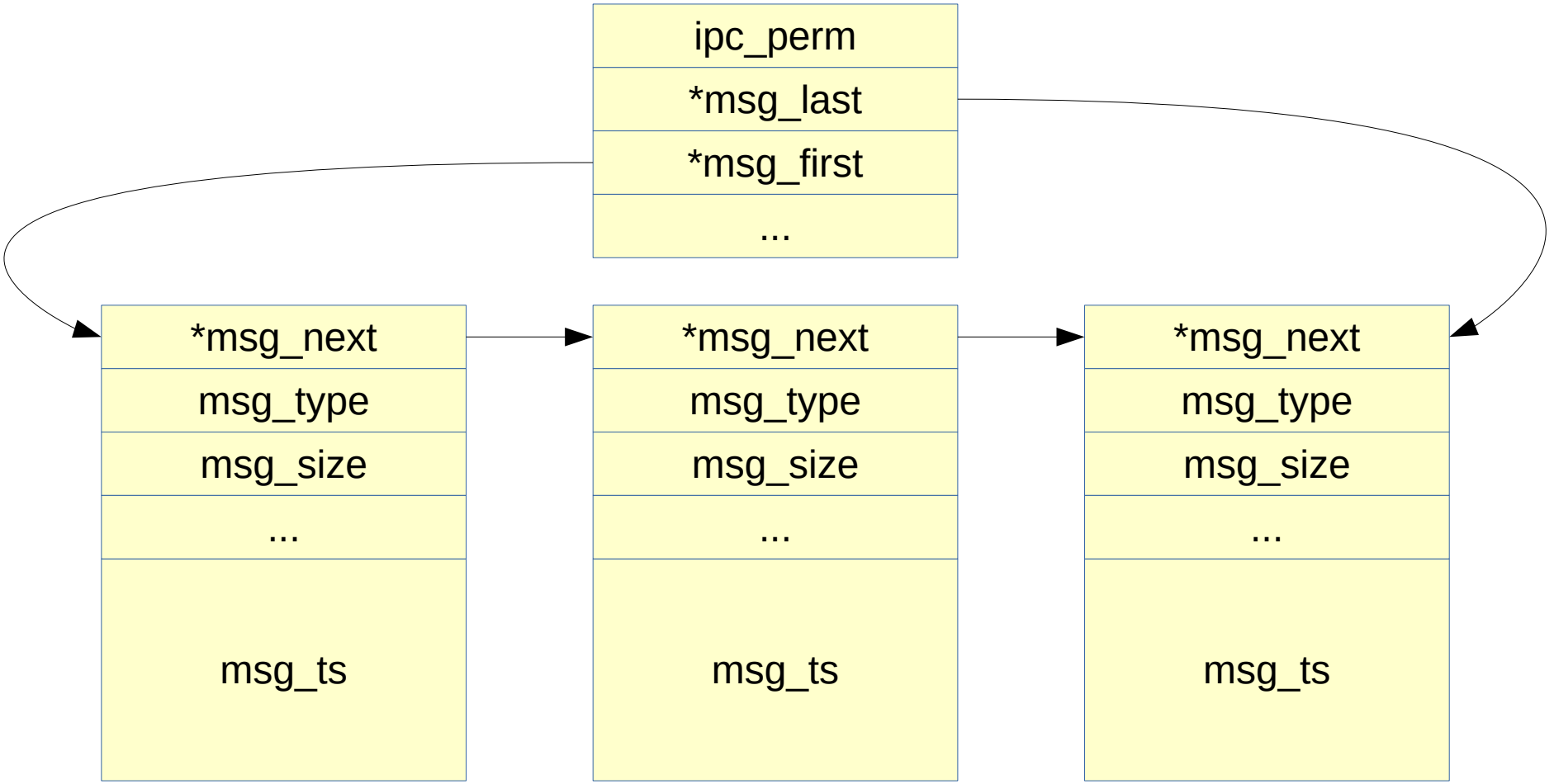
Message Queue (1)

- “Message queues can be best described as an internal linked list within the kernel's addressing space. Messages can be sent to the queue in order and retrieved from the queue in several different ways. Each message queue (of course) is uniquely identified by an IPC identifier.”[1]

[1] The Linux Documentation Project, http://www.tldp.org

Andre M. Maier
Dept. of Computer Science

Message Queue (2)

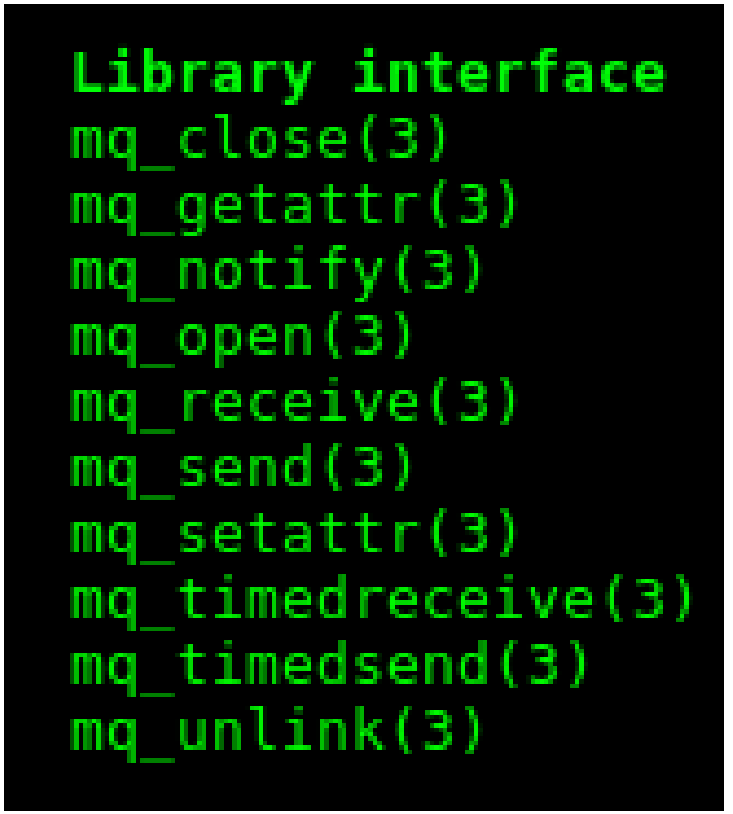


DHBW
Duale Hochschule
Baden-Württemberg

Andre M. Maier
Dept. of Computer Science

Message Queue (3)

- Linux Implementation
  - `man 7 mq_overview`
  - Message queue files are virtual single inodes mounted under `/dev/mqueue`
  - Programs that use the message queue API must be linked against the real-time library `librt` (compiler option `-lrt`)



DHBW
Duale Hochschule
Baden-Württemberg

Andre M. Maier
Dept. of Computer Science

Synchronization

- Common problems in IPC
  - Critical Section Problem
  - Bounded-Buffer Problem
  - Dining Philosophers Problem
  - Readers-Writers Problem
  - Sleeping Barber Problem
  - Cigarette Smokers Problem

Andre M. Maier
Dept. of Computer Science

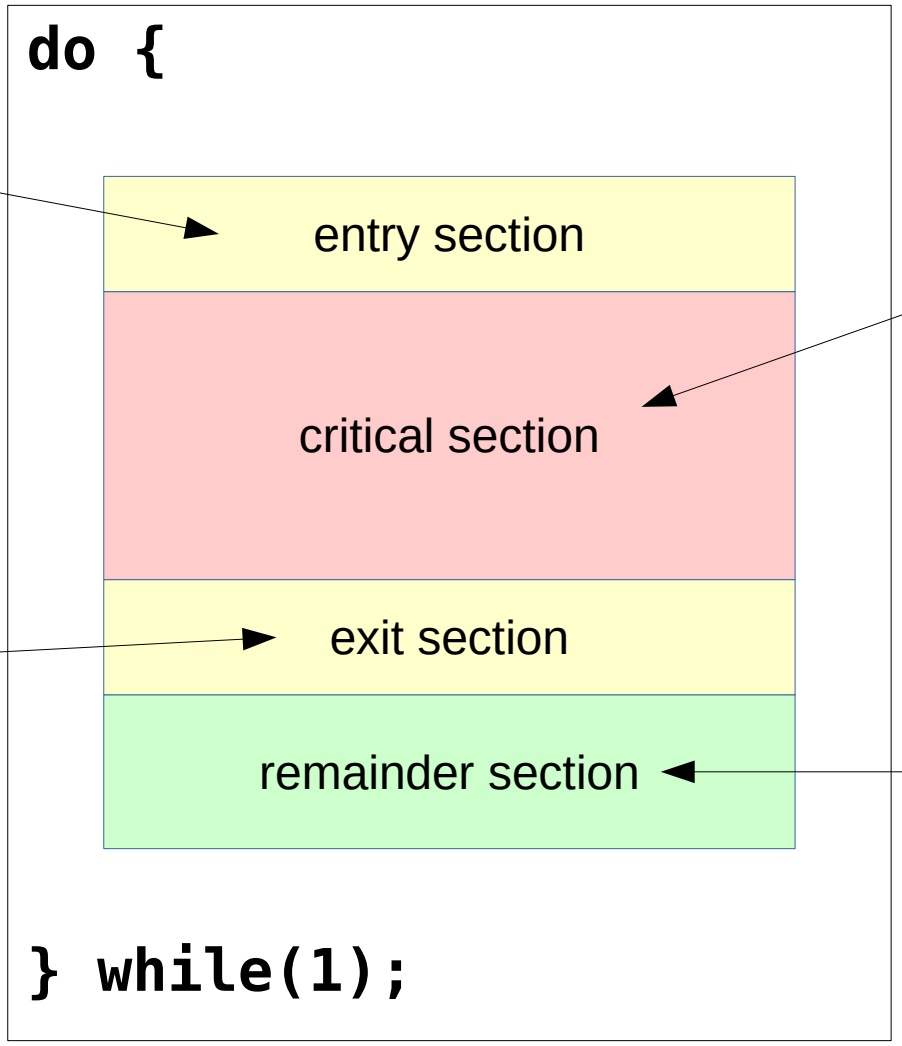
Critical Section Problem (1)

- Code segment that access shared variables have to be executed as atomic actions.
- Such code segments are called *critical sections*.
- Only one process shall be allowed to execute a critical section at a time.
- Any other process will have to wait before entering the critical section.

Andre M. Maier
Dept. of Computer Science

Critical Section Problem (2)

Waits for the resources to become available acquires a lock on the required resources.



Critical code to be executed as an atomic action

Releases the lock from the resources and notifies others the critical section is over.

Non-critical code

DHBW
Duale Hochschule
Baden-Württemberg

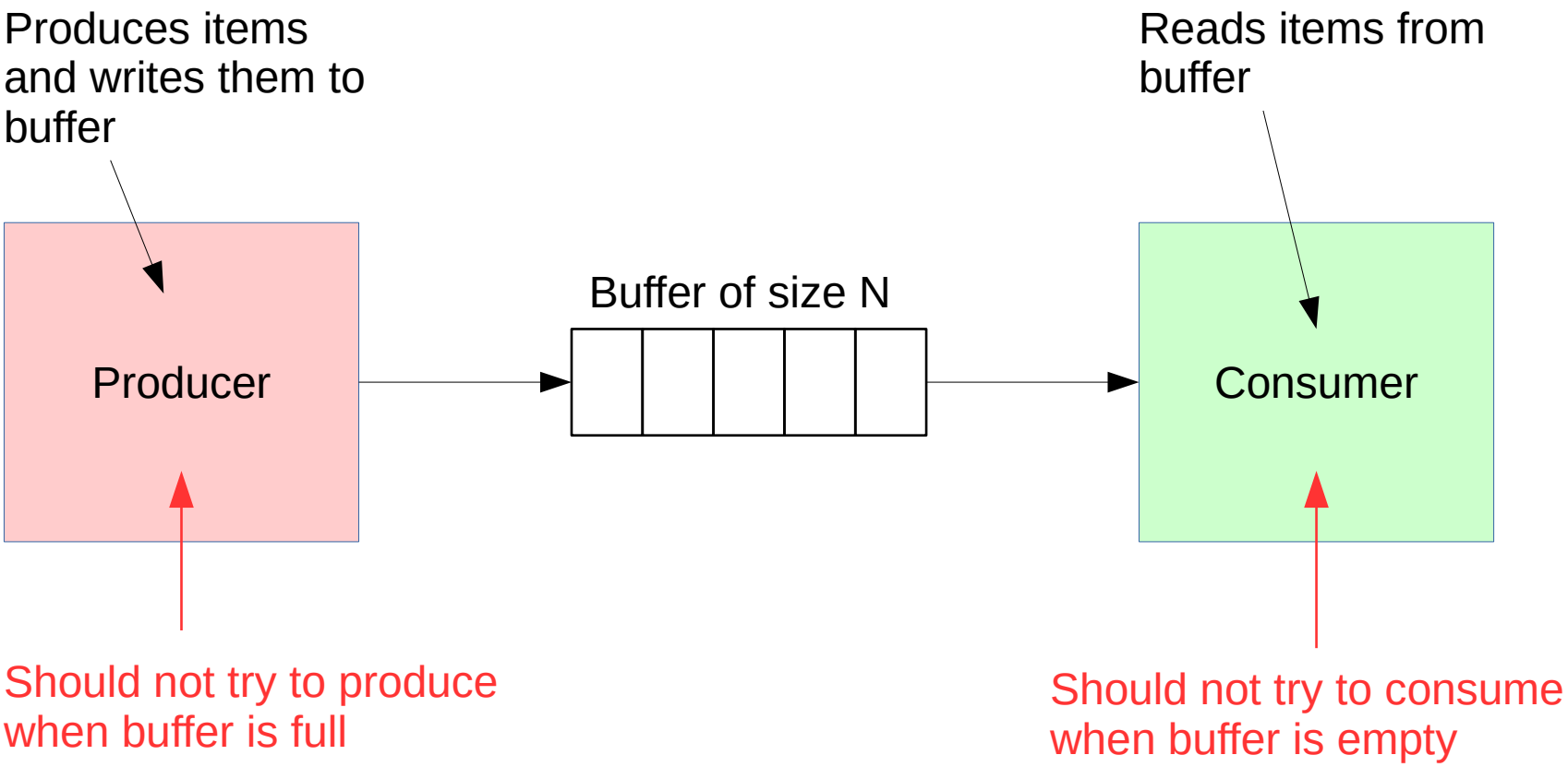
Andre M. Maier
Dept. of Computer Science

Critical Sections Problem (3)

- The following conditions must be satisfied to solve the problem.
  - Mutual exclusion
    - Only one process may be inside the critical section at any time. All other processes have to wait until the critical section is over.
  - Progress
    - If no process is currently executing the critical section, the selection of the process that will enter the critical section next must not be postponed indefinitely; assuming that there are processes that wish to enter the critical section.
  - Bounded waiting
    - The waiting time after a process has made a request to enter the critical section and before the request is granted is limited. In other words, the number of times that other processes are allowed to enter the critical section during that interval must be limited.

Andre M. Maier
Dept. of Computer Science

Bounded-Buffer Problem



DHBW
Duale Hochschule Baden-Württemberg
Andre M. Maier
Dept. of Computer Science

Assignment

- Study and explain the following synchronization methods:
  - Spinlock
  - Semaphore
  - Mutual Exclusion
  - Monitor
- Explain how to solve the bounded-buffer problem.

Andre M. Maier
Dept. of Computer Science

Lab Exercises

- Write two C programs that send exchange signals (ping and response) every second.
- Write two programs that exchange information via message queue.
- Implement a simulation of the producer consumer problem including a solution.
- Write a client and a server that communicate via a Unix domain socket.

Andre M. Maier
Dept. of Computer Science

Questions for Review

- What is the difference between independent processes and cooperating processes?
- Describe what happens if a process in blocked state receives a signal.
- What is the difference between a pipe and a named pipe?
- Draw a diagram that illustrates the principle of indirect message passing.
- What kind of data structure is a Unix message queue based on?
- Explain the producer-consumer problem.
- What is the main disadvantage of spinlocks?
- What is the difference between a mutex and a semaphore?

Andre M. Maier
Dept. of Computer Science