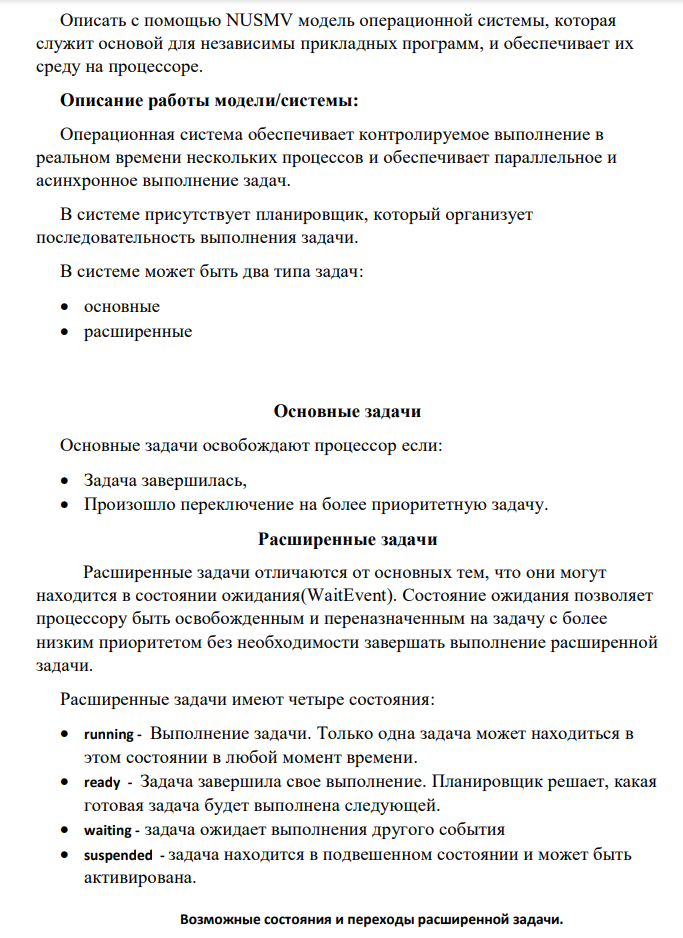
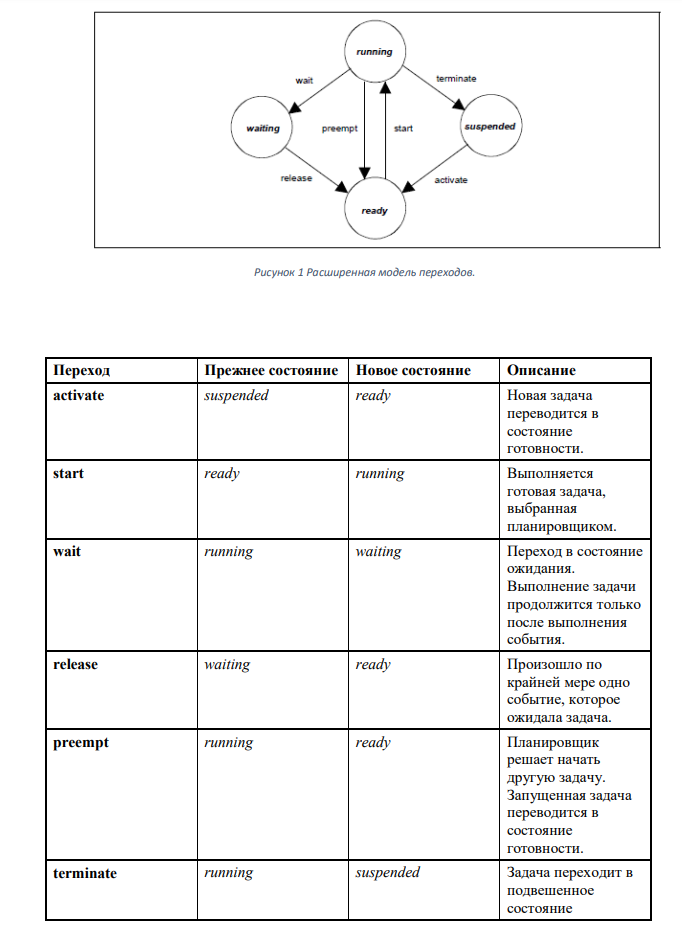
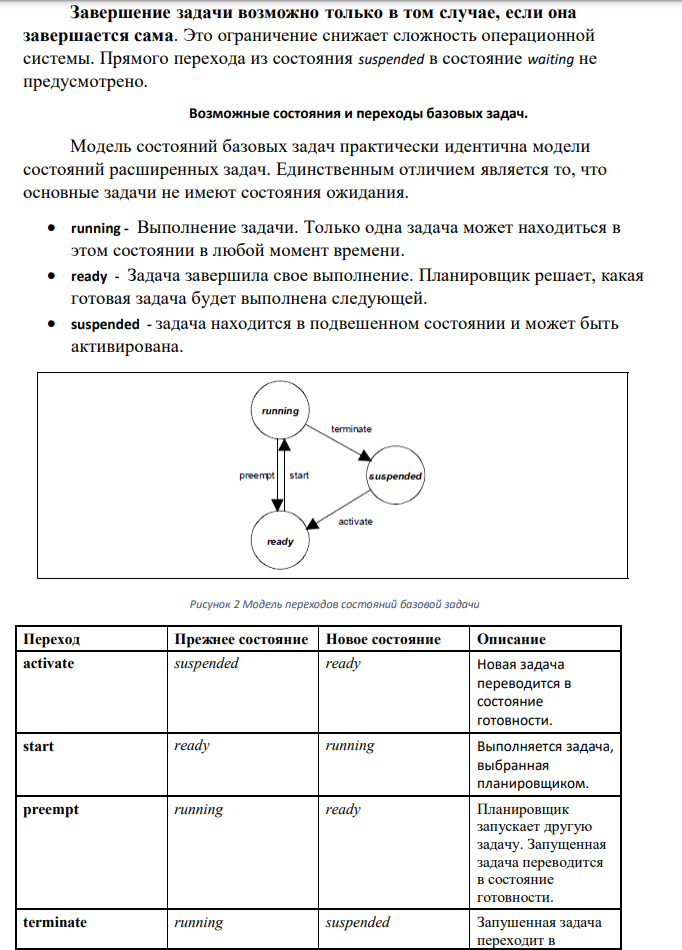
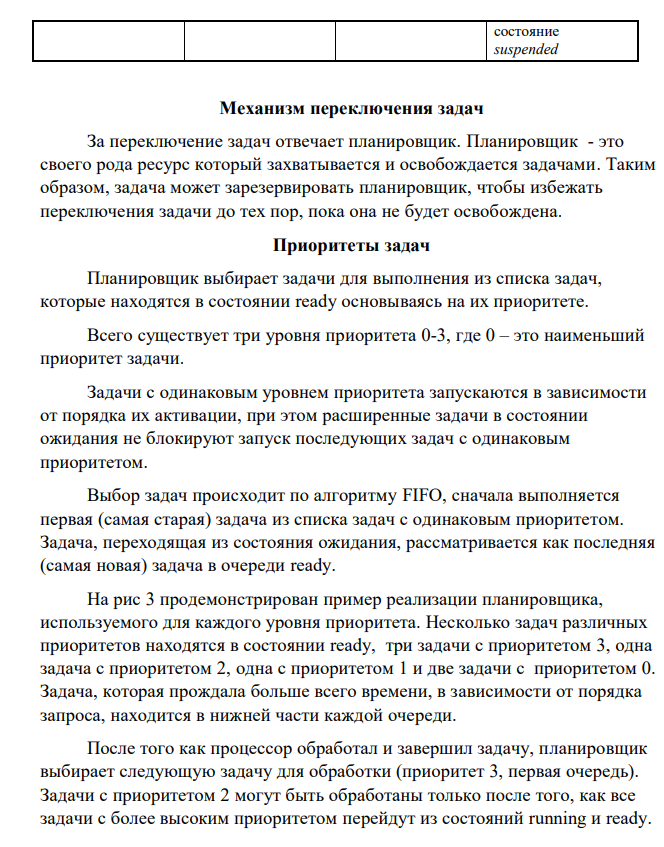
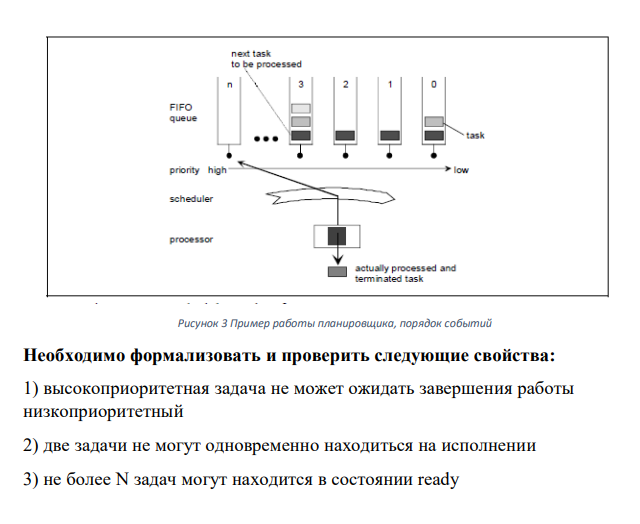
**Постановка задачи**











**Решение**

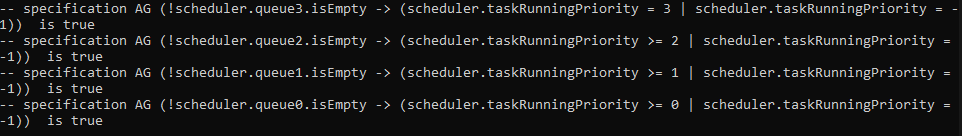
1. Высокоприоритетная задача не может ожидать завершения работы низкоприоритетный

CTLSPEC AG ((!scheduler.queue3.isEmpty) -> (scheduler.taskRunningPriority = 3 | scheduler.taskRunningPriority = -1));

CTLSPEC AG ((!scheduler.queue2.isEmpty) -> (scheduler.taskRunningPriority >= 2 | scheduler.taskRunningPriority = -1));

CTLSPEC AG ((!scheduler.queue1.isEmpty) -> (scheduler.taskRunningPriority >= 1 | scheduler.taskRunningPriority = -1));

CTLSPEC AG ((!scheduler.queue0.isEmpty) -> (scheduler.taskRunningPriority >= 0 | scheduler.taskRunningPriority = -1));



1. Две задачи не могут одновременно находиться на исполнении

CTLSPEC AG ((scheduler.task1.state = RUNNING) -> (scheduler.task2.state != RUNNING &

scheduler.task3.state != RUNNING &

scheduler.task4.state != RUNNING &

scheduler.task5.state != RUNNING));

CTLSPEC AG ((scheduler.task2.state = RUNNING) -> (scheduler.task1.state != RUNNING &

scheduler.task3.state != RUNNING &

scheduler.task4.state != RUNNING &

scheduler.task5.state != RUNNING));

CTLSPEC AG ((scheduler.task3.state = RUNNING) -> (scheduler.task1.state != RUNNING &

scheduler.task2.state != RUNNING &

scheduler.task4.state != RUNNING &

scheduler.task5.state != RUNNING));

CTLSPEC AG ((scheduler.task4.state = RUNNING) -> (scheduler.task1.state != RUNNING &

scheduler.task2.state != RUNNING &

scheduler.task3.state != RUNNING &

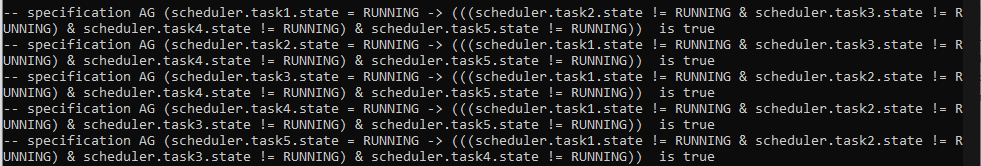
scheduler.task5.state != RUNNING));

CTLSPEC AG ((scheduler.task5.state = RUNNING) -> (scheduler.task1.state != RUNNING &

scheduler.task2.state != RUNNING &

scheduler.task3.state != RUNNING &

scheduler.task4.state != RUNNING));



1. Не более N задач могут находиться в состоянии ready

CTLSPEC AG (scheduler.readyTasksCount <= N);



Все наши верификации выполняются

**Программа**

MODULE ExtendedTask(id, priority)

VAR

state: { RUNNING, READY, WAITING, SUSPENDED };

command: { wait, terminate, preempt, activate, release, start, nop };

ASSIGN

init(state) := SUSPENDED;

init(command) := nop;

ASSIGN

next(state) :=

case

command = activate : READY;

command = start : RUNNING;

command = preempt : READY;

command = wait : WAITING;

command = terminate : SUSPENDED;

command = release : READY;

TRUE : state;

esac;

MODULE BasicTask(id, priority)

VAR

state: { RUNNING, READY, SUSPENDED };

command: { preempt, terminate, activate, start, nop };

ASSIGN

init(state) := SUSPENDED;

init(command) := nop;

ASSIGN

next(state) :=

case

command = activate : READY;

command = start : RUNNING;

command = preempt : READY;

command = terminate : SUSPENDED;

TRUE : state;

esac;

MODULE FIFO(command, taskId)

VAR

queue: array 1..3 of 0..4;

ASSIGN

init(queue[1]) := 0;

init(queue[2]) := 0;

init(queue[3]) := 0;

DEFINE

queueSize := count(queue[1] != 0, queue[2] != 0, queue[3] != 0);

firstTask := queue[1];

isEmpty := queueSize = 0;

isFull := queueSize = 3;

TRANS

case

(command = DEQUEUE) : next(queue[1]) = queue[2] & next(queue[2]) = queue[3] & next(queue[3]) = 0;

(command = ENQUEUE):

case

(queueSize = 0) : (next(queue[1]) = taskId & next(queue[2]) = queue[2] & next(queue[3]) = queue[3]);

(queueSize = 1) : (next(queue[1]) = queue[1] & next(queue[2]) = taskId & next(queue[3]) = queue[3]);

(queueSize = 2) : (next(queue[1]) = queue[1] & next(queue[2]) = queue[2] & next(queue[3]) = taskId);

TRUE : (next(queue[1]) = queue[1] & next(queue[2]) = queue[2] & next(queue[3]) = queue[3]);

esac;

TRUE : (next(queue[1]) = queue[1] & next(queue[2]) = queue[2] & next(queue[3]) = queue[3]);

esac;

MODULE Scheduler(maxTasksCount)

VAR

taskRunningId: -1..4;

taskRunningPriority: -1..3;

task1: BasicTask(0, 0);

task2: ExtendedTask(1, 1);

task3: BasicTask(2, 1);

task4: ExtendedTask(3, 2);

task5: BasicTask(4, 3);

command0: { ENQUEUE, DEQUEUE, NOP };

command1: { ENQUEUE, DEQUEUE, NOP };

command2: { ENQUEUE, DEQUEUE, NOP };

command3: { ENQUEUE, DEQUEUE, NOP };

taskIdAdded0: -1..4;

taskIdAdded1: -1..4;

taskIdAdded2: -1..4;

taskIdAdded3: -1..4;

queue0: FIFO(command0, taskIdAdded0);

queue1: FIFO(command1, taskIdAdded1);

queue2: FIFO(command2, taskIdAdded2);

queue3: FIFO(command3, taskIdAdded3);

ASSIGN

init(command0) := NOP;

init(command1) := NOP;

init(command2) := NOP;

init(command3) := NOP;

init(taskIdAdded0) := -1;

init(taskIdAdded1) := -1;

init(taskIdAdded2) := -1;

init(taskIdAdded3) := -1;

init(taskRunningId) := -1;

init(taskRunningPriority) := -1;

DEFINE

readyTasksCount := count(task1.state = READY,

task2.state = READY,

task3.state = READY,

task4.state = READY,

task5.state = READY);

enoughSpaceQueue0 := !queue0.isFull & readyTasksCount < maxTasksCount;

enoughSpaceQueue1 := !queue1.isFull & readyTasksCount < maxTasksCount;

enoughSpaceQueue2 := !queue2.isFull & readyTasksCount < maxTasksCount;

enoughSpaceQueue3 := !queue3.isFull & readyTasksCount < maxTasksCount;

ASSIGN

next(taskRunningPriority) :=

case

next(task1.command) = start : 0;

next(task2.command) = start : 1;

next(task3.command) = start : 1;

next(task4.command) = start : 2;

next(task5.command) = start : 3;

TRUE : taskRunningPriority;

esac;

-- Task 1

ASSIGN

next(task1.command) :=

case

(task1.state = SUSPENDED & enoughSpaceQueue0) : { activate, nop };

(task1.state = SUSPENDED & !enoughSpaceQueue0) : nop;

(task1.state = READY & taskRunningId = -1 & !queue0.isEmpty & queue1.isEmpty & queue2.isEmpty & queue3.isEmpty & task1.id = queue0.firstTask) : start;

(task1.state = RUNNING & enoughSpaceQueue0) : { preempt, terminate, nop };

(task1.state = RUNNING & !enoughSpaceQueue0) : { terminate, nop };

TRUE : nop;

esac;

TRANS

case

(next(task1.command) = activate | next(task1.command) = preempt) : (next(command0) = ENQUEUE & next(taskIdAdded0) = task1.id);

(next(task1.command) = start) : (next(taskRunningId) = task1.id & next(command0) = DEQUEUE);

(next(task1.command) = terminate) : (next(taskRunningId) = -1 & next(command0) = NOP);

TRUE : (next(taskRunningId) = taskRunningId & next(command0) = NOP);

esac;

-- Task 2

ASSIGN

next(task2.command) :=

case

(task2.state = SUSPENDED & enoughSpaceQueue1) : { activate, nop };

(task2.state = SUSPENDED & !enoughSpaceQueue1) : nop;

(task2.state = READY & taskRunningId = -1 & !queue1.isEmpty & queue2.isEmpty & queue3.isEmpty & task2.id = queue1.firstTask) : start;

(task2.state = RUNNING & enoughSpaceQueue1) : { preempt, terminate, wait, nop };

(task2.state = RUNNING & !enoughSpaceQueue1) : { terminate, wait, nop };

(task2.state = WAITING & enoughSpaceQueue1) : release;

TRUE : nop;

esac;

TRANS

case

(next(task2.command) = activate | next(task2.command) = release | next(task2.command) = preempt) : (next(command1) = ENQUEUE & next(taskIdAdded1) = task2.id);

(next(task2.command) = start) : (next(taskRunningId) = task2.id & next(command1) = DEQUEUE);

(next(task2.command) = terminate) : (next(taskRunningId) = -1 & next(command1) = NOP);

TRUE : (next(taskRunningId) = taskRunningId & next(command1) = NOP);

esac;

-- Task 3

ASSIGN

next(task3.command) :=

case

(task3.state = SUSPENDED & enoughSpaceQueue1) : { activate, nop };

(task3.state = SUSPENDED & !enoughSpaceQueue1) : nop;

(task3.state = READY & taskRunningId = -1 & queue3.isEmpty & queue2.isEmpty & !queue1.isEmpty & task3.id = queue0.firstTask) : start;

(task3.state = RUNNING & enoughSpaceQueue1) : { preempt, terminate, nop };

(task3.state = RUNNING & !enoughSpaceQueue1) : { terminate, nop };

TRUE : nop;

esac;

TRANS

case

(next(task3.command) = activate | next(task3.command) = preempt) : (next(command1) = ENQUEUE & next(taskIdAdded1) = task3.id);

(next(task3.command) = start) : (next(taskRunningId) = task3.id & next(command1) = DEQUEUE);

(next(task3.command) = terminate) : (next(taskRunningId) = -1 & next(command1) = NOP);

TRUE : (next(taskRunningId) = taskRunningId & next(command1) = NOP);

esac;

-- Task 4

ASSIGN

next(task4.command) :=

case

(task4.state = SUSPENDED & enoughSpaceQueue2) : { activate, nop };

(task4.state = SUSPENDED & !enoughSpaceQueue2) : nop;

(task4.state = READY & taskRunningId = -1 & queue3.isEmpty & !queue2.isEmpty & task4.id = queue1.firstTask) : start;

(task4.state = RUNNING & enoughSpaceQueue2) : { preempt, terminate, wait, nop };

(task4.state = RUNNING & !enoughSpaceQueue2) : { terminate, wait, nop };

(task4.state = WAITING & enoughSpaceQueue2) : release;

TRUE : nop;

esac;

TRANS

case

(next(task4.command) = activate | next(task4.command) = release | next(task4.command) = preempt) : (next(command2) = ENQUEUE & next(taskIdAdded2) = task4.id);

(next(task4.command) = start) : (next(taskRunningId) = task4.id & next(command2) = DEQUEUE);

(next(task4.command) = terminate) : (next(taskRunningId) = -1 & next(command2) = NOP);

TRUE : (next(taskRunningId) = taskRunningId & next(command2) = NOP);

esac;

-- Task 5

ASSIGN

next(task5.command) :=

case

(task5.state = SUSPENDED & enoughSpaceQueue3) : { activate, nop };

(task5.state = SUSPENDED & !enoughSpaceQueue3) : nop;

(task5.state = READY & taskRunningId = -1 & !queue3.isEmpty & task5.id = queue0.firstTask) : start;

(task5.state = RUNNING & enoughSpaceQueue3) : { preempt, terminate, nop };

(task5.state = RUNNING & !enoughSpaceQueue3) : { terminate, nop };

TRUE : nop;

esac;

TRANS

case

(next(task5.command) = activate | next(task5.command) = preempt) : (next(command3) = ENQUEUE & next(taskIdAdded3) = task5.id);

(next(task5.command) = start) : (next(taskRunningId) = task5.id & next(command3) = DEQUEUE);

(next(task5.command) = terminate) : (next(taskRunningId) = -1 & next(command3) = NOP);

TRUE : (next(taskRunningId) = taskRunningId & next(command3) = NOP);

esac;

MODULE main

DEFINE

N := 5;

VAR

scheduler: Scheduler(N);

-- "Высокоприоритетная задача не может ожидать завершения работы низкоприоритетной"

CTLSPEC AG ((!scheduler.queue3.isEmpty) -> (scheduler.taskRunningPriority = 3 | scheduler.taskRunningPriority = -1));

CTLSPEC AG ((!scheduler.queue2.isEmpty) -> (scheduler.taskRunningPriority >= 2 | scheduler.taskRunningPriority = -1));

CTLSPEC AG ((!scheduler.queue1.isEmpty) -> (scheduler.taskRunningPriority >= 1 | scheduler.taskRunningPriority = -1));

CTLSPEC AG ((!scheduler.queue0.isEmpty) -> (scheduler.taskRunningPriority >= 0 | scheduler.taskRunningPriority = -1));

-- "Две задачи не могут одновременно находиться на исполнении"

CTLSPEC AG ((scheduler.task1.state = RUNNING) -> (scheduler.task2.state != RUNNING &

scheduler.task3.state != RUNNING &

scheduler.task4.state != RUNNING &

scheduler.task5.state != RUNNING));

CTLSPEC AG ((scheduler.task2.state = RUNNING) -> (scheduler.task1.state != RUNNING &

scheduler.task3.state != RUNNING &

scheduler.task4.state != RUNNING &

scheduler.task5.state != RUNNING));

CTLSPEC AG ((scheduler.task3.state = RUNNING) -> (scheduler.task1.state != RUNNING &

scheduler.task2.state != RUNNING &

scheduler.task4.state != RUNNING &

scheduler.task5.state != RUNNING));

CTLSPEC AG ((scheduler.task4.state = RUNNING) -> (scheduler.task1.state != RUNNING &

scheduler.task2.state != RUNNING &

scheduler.task3.state != RUNNING &

scheduler.task5.state != RUNNING));

CTLSPEC AG ((scheduler.task5.state = RUNNING) -> (scheduler.task1.state != RUNNING &

scheduler.task2.state != RUNNING &

scheduler.task3.state != RUNNING &

scheduler.task4.state != RUNNING));

-- "Не более N задач могут находиться в состоянии ready"

CTLSPEC AG (scheduler.readyTasksCount <= N);