**Краткие методические рекомендации преподавателям по
дисциплине**

«Многопоточное и распределенное программирование»

Разделы дисциплины «Многопоточное и распределенное программирование»

Всего 3 темы лекций:

- 1) Работа с потоками
- 2) Параллельное программирование
- 3) Управление памятью в многопоточных системах

9 практических занятий

3 коллоквиума

Методика расчета оценки текущего контроля и итоговой оценки

Допуск к зачету с оценкой получают студенты, посетившие не менее 75% всех видов занятий, успешно (не менее чем на оценку «Удовлетворительно») прошедшие контрольные точки текущего контроля в соответствии с графиком текущего контроля успеваемости (3 коллоквиума).

На коллоквиуме студент должен развернуто ответить на один вопрос.

Критерии оценивания:

«отлично» - вопрос раскрыт полностью, задача решена правильно

«хорошо» - вопрос раскрыт не полностью, задача решена частично

«удовлетворительно» - в ответе на вопрос имеются существенные ошибки; задача не решена или решена неправильно, ход решения правильный

«неудовлетворительно» - отсутствует ответ на вопрос или содержание ответа не совпадает с поставленным вопросом, задача не решена, ход решения неправильный

Зачет с оценкой проводится по билетам. Критерии оценивания приведены в п. 6.1 Рабочей программы дисциплины.

Литература

Основная литература

1) Дорогов, Александр Юрьевич. Синхронизация и взаимодействие программных потоков в операционной среде реального времени [Текст]: учеб. пособие / А.Ю. Дорогов, 2007. 64 с

2) Таненбаум Э. Современные операционные системы. 4е изд. [Электронный ресурс] / Э. Таненбаум, Х. Бос, 2019. 1120 с

Дополнительная литература

1) Бабичев, Сергей Леонидович. Распределенные системы [Электронный ресурс]: Учебное пособие для вузов / Бабичев С. Л., Коньков К. А., 2020. 507 с

Проверка базовых знаний

1. Расположите в порядке увеличения степени абстракции технологии высокопроизводительных вычислений:

- OS Threads, SSE/MMX, C++11 Threads/Java Threads, TBB/Fork-Join framework
- SSE/MMX, OS Threads, TBB/Fork-Join framework, C++11 Threads/Java Threads
- **SSE/MMX, OS Threads, C++11 Threads/Java Threads, TBB/Fork-Join framework**
- OS Threads, TBB/Fork-Join framework, C++11 Threads/Java Threads, SSE/M

2. Что показывает закон Амдала:

• **Прирост производительности в зависимости от доли последовательного кода и числа вычислительных элементов**

• Прирост производительности в зависимости от числа вычислительных элементов и используемой технологии распараллеливания

• Число вычислительных элементов, необходимое для достижения заданного уровня роста производительности

• Долю последовательного кода, производительность которого невозможно повысить

3. Как называются функции, внутри которых, в зависимости от используемой технологии, происходит проверка на необходимость завершения работы потока и по результатам проверки возможно выкидывание исключений или завершение потока с применением стека зарегистрированных функций:

- Функции завершения потока
- join-функции
- **Cancellation / interruption points**
- Signal handlers

4. Чем отличается spin mutex от обычного mutex:

- **Остаётся всегда в user space**
- Остаётся всегда в kernel space
- Ничем не отличается
- Позволяет читателям производить операции параллельно