- а) Верно ли утверждение «Если для двух процессов не выполнены условия Бернстайна, то они образуют недетерминированный набор»? Если Ваш ответ «да» докажите, если «нет» приведите контрпример.
- b) Для чего в мониторах Хора вводятся условные переменные? Приведите пример их использования.
- с) В чем заключается разница между процессами и нитями исполнения (thread'aми), реализованными на уровне ядра ОС?

Решение:

- а) Утверждение неверно. Рассмотрим набор из двух процессов с одинаковым кодом из двух атомарных операций: x = 2; y = x+3; Все условия Бернстайна нарушены набор, тем не менее, детерминирован.
- b) Условные переменные в мониторах Хора, как правило, применяются для взаимной синхронизации процессов, требующейся для обеспечения правильной очередности их доступа к разделяемым ресурсам (поскольку взаимоисключения обеспечивает сама конструкция мониторов). Пример ну, скажем, см. в лекциях решение задачи producer-consumer.
- с) В системах, поддерживающих нити исполнения на уровне ядра, thread'ы представляют собой единицы исполнения, а процессы единицы выделения ресурсов. Процесс представляется как совокупность взаимодействующих нитей и выделенных ему ресурсов. Нити процесса разделяют его программный код, глобальные переменные и системные ресурсы, но каждая нить имеет свой собственный программный счетчик, свое содержимое регистров и свой собственный стек. Планирование использования процессора происходит в терминах нитей, а управление памятью и другими системными ресурсами остается в терминах процессов.
- 1. (6 баллов) Рассмотрим однопроцессорную одноядерную вычислительную систему с объемом свободной оперативной памяти 220 Мb, в которой используется схема организации памяти с динамическими (переменными) разделами. Для долгосрочного планирования процессов в ней применен алгоритм планирования с приоритетами. В систему поступают пять заданий с различной длительностью, различным объемом занимаемой памяти и различными приоритетами по следующей схеме:

| Номер задания | Момент поступления в очередь заданий | Время исполнения (CPU burst) | Объем занимаемой памяти | Приоритет |
|------------------|--------------------------------------|------------------------------|-------------------------------|-----------|
| 1 | 0 | 6 | 60 Mb | 2 |
| 2 | 0 | 3 | 100 Mb | 1 |
| 3 | 3 | 4 | 60 Mb | 3 |
| 4 | 4 | 3 | 80 Mb | 1 |
| 5 | 5 | 2 | 20 Mb | 2 |

Нарисуйте диаграмму использования оперативной памяти, вычислите среднее время между стартом задания и его завершением (turnaround time) и среднее время ожидания (waiting time) для следующей комбинации алгоритма краткосрочного планирования и стратегии размещения процессов в памяти:

Вытесняющий SJF (Shortest Job First) и worst fit (наименее подходящий);

При вычислениях считать, что процессы не совершают операций ввода-вывода. Временами переключения контекста, рождения процессов и работы алгоритмов планирования пренебречь. Освобождение памяти, занятой процессами, происходит немедленно по истечении их CPU burst. Краткосрочное планирование осуществляется после рождения новых процессов в текущий момент времени. Наивысшим приоритетом в системе является приоритет со значением 0.

Решение:

Рассмотрим выполнение процессов в системе для вытесняющего алгоритма SJF и стратегии worst fit. По вертикали в таблице отложены номера процессов, по горизонтали — промежутки времени. Столбец 0 соответствует временному интервалу от 0 до 1. Буква «И» означает состояние исполнения, буква «Г» — состояние готовности, буква «О» — ожидание в очереди заданий. Под таблицей приведено распределение памяти, а еще ниже — содержимое очереди заданий.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|
| 1 | Γ | Γ | Γ | Γ | Γ | Γ | Γ | Γ | Γ | Γ | Γ | Γ | И | И | И | И | И | И |
| 2 | И | И | И | | | | | | | | | | | | | | | |
| 3 | | | | И | И | И | И | | | | | | | | | | | |
| 4 | | | | | О | О | О | Γ | Γ | И | И | И | | | | | | |
| 5 | | | | | | О | О | И | И | | | | | | | | | |

| D2 | D2 | D2 | P3 60 | P3 60 | P3 60 | P3 60 | P4 80 | P4 80 | P4 80 | P4 80 | P4 80 | 100 | 100 | 100 | 100 | 100 | 100 |
|-----------|-----------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-------|----------|----------|
| P2 100 | P2 100 | P2 100 | 40 | 40 | 40 | 40 | 20 | 20 | 20 | 20 | 20 | 100 | 100 | 100 | 100 | 100 | 100 |
| P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 | P1 60 |
| | | | | | | | P5 20 | P5 20 | | | | | | | | | |
| 60 | 60 | 60 | 60 | 60 | 60 | 60 | 40 | 40 | 60 | 60 | 60 | 60 | 60 | 60 | 60 | 60 | 60 |

| † | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|----------|---|---|---|---|----|----|----|---|---|---|----|----|----|----|----|----|----|----|
| | | | | | P4 | P4 | P4 | | | | | | | | | | | |
| 1 | | | | | | P5 | P5 | | | | | | | | | | | |

Среднее время между стартом задания и его завершением: tt = (18 + 3 + 4 + 8 + 4)/5 = 7.4Среднее время ожидания: wt = (12 + 0 + 0 + 5 + 2)/5 = 3.8.

(3 балла) В вычислительной системе с сегментно-страничной организацией памяти и 32-х битовым адресом максимальный размер сегмента составляет 8 Мb, а размер страницы памяти 512 Кb. Для некоторого процесса в этой системе таблица сегментов имеет вид:

| Номер сегмента | Длина сегмента |
|----------------|----------------|
| 0 | 0x500000 |
| 1 | 0x3e0000 |

Таблицы страниц, находящихся в памяти, для сегментов 0 и 1 приведены ниже:

| C | • |
|---------|---|
| Сегмент | ı |

| Номер страницы | Номер кадра (десятичный) |
|----------------|--------------------------|
| 0 | 19 |
| 3 | 7 |
| 6 | 0 |

| Сегмент | |
|---------|--|
| | |

| Номер страницы | Номер кадра (десятичный) |
|----------------|--------------------------|
| 1 | 12 |
| 3 | 63 |
| 7 | 32 |

Каким физическим адресам соответствуют логические адреса: 0x00ac3145, 0x00d17281, 0x009a1530?

Решение:

8 Mb — это 2^{23} байт, т.е. под номер сегмента в логическом адресе отводится 9 бит, a 23 бита — под смещение внутри сегмента. Размер страницы 512 Kb — это 2¹⁹ байт, т.е. из смещения внугри сегмента 19 бит отводится под смещение внугри страницы, а 4 бита — под

0x00ac3145 —> сегмент 1, смещение 0x2c3145 —> сегмент 1, страница 5, смещение 0x00043145 —> page fault. 0x00d17281 —> сегмент 1, смещение 0x517281 —> смещение больше размеров сегмента —> segmentation fault.

0х009а1530 —> сегмент 1, смещение 0х001а1530 —> сегмент 1, страница 3, смещение 0х00021530 —>

кадр 63, смещение $0x00021530 \longrightarrow$ кадр 0x3f, смещение $0x00021530 \longrightarrow 0x01f80000+0x00021530 \longrightarrow$ **0x01fa1530**.

(12 баллов) На Урале в карьере открытым способом ведется добыча породы, содержащей железную руду. Самосвалы отвозят породу с рудой на обогатительную фабрику, а обратно с фабрики на отвалы карьера везут отработанную пустую породу. На дороге между карьером и фабрикой находится узкий хлипкий мост. Машины, едущие в одном направлении,

могут переезжать мост одновременно, а в противоположных — нет. Мост выдерживает не более двух машин. Машины, ждущие въезда на мост, начинают по нему движение, как только появляется такая возможность. Используя классические мониторы Хора, постройте корректную модель проезда через мост, описав движение каждого самосвала с помощью отдельного процесса.

Возможное решение:

```
Считаем, что мост расположен с севера на юг.
    monitor Bridge {
        condition N, S;
                                       // условные переменные для ожидания въезда на мост с севера и с юга
        int Count, Dir;
                                       // количество машин на мосту и направление движения: 1 или 2
        void north_entry(void){
                                       // въезд с севера
         if (Count == 2 \parallel (Count == 1 \&\& Dir != 1) N.wait;
         Count++; Dir = 1;
         if (Count == 1) N.signal;
        void south_entry(void) {
                                       // въезд с юга
         if (Count == 2 \parallel (Count == 1 \&\& Dir != 2) S.wait;
         Count++; Dir = 2;
         if(Count == 1) S.signal;
        void north_exit(void) {
                                       // выезд с севера на юг
         Count--:
         if (Count == 1) N.signal;
         else if (Count == 0) S.signal;
        void south_exit(void) {
                                       // выезд с юга на север
         Count--;
         if (Count == 1) S.signal;
         else if (Count == 0) N.signal;
         Count = Dir = 0;
Процесс самосвал:
    while(1) {
        Загрузиться на карьере.
        Доехать до моста.
        Bridge.north_entry();
        Проехать мост
        Bridge.north_exit();
        Выгрузка и погрузка на фабрике.
        Доехать до моста.
        Bridge.south_entry();
        Проехать мост
        Bridge.south_exit();
        Выгрузка на карьере.
    }
```

Оценка:

Грубые ошибки: тупиковые ситуации, лобовое столкновение машин, мост рухнул — -9 баллов. Ошибки средней тяжести: циклы активного ожидания, сначала все едут в одну сторону, потом – все в другую — -6 баллов Полный балл только за полностью правильный ответ. Любая попытка решения должна быть оценена.