Сесия 1 за дисциплина “Операционни системи”, СУ, ФМИ, 28.06.2020 г.

Теоретични задачи за СИ:

Задача 1. Множество паралелно работещи копия на всеки от процесите `P` и `Q` изпълняват поредица от две инструкции:

```
process P                process Q
  p_1                      q_1
  p_2                      q_2
```

Осигурете чрез семафори синхронизация на `P` и `Q`, така че поне една инструкция `p_1` да се изпълни преди всички `q_2`, и поне една инструкция `q_1` да се изпълни преди всички `p_2`.

Задача 2. Опишете накратко основните комуникационни канали в ОС Linux.
Кои канали използват пространството на имената и кои не го правят?

Примерни решения

Задача 1. За двете искани в условието синхронизации използваме два семафора – `t1` и `t2`, инициализираме ги с блокиращо начално състяние:

```
semaphore t1,t2
t1.init(0)
t2.init(0)
```

Добавяме в кода на процесите `P` и `Q` синхронизиращи инструкции:

```
process P                  process Q
  p_1                        q_1
  t1.signal()                t2.signal()
  t2.wait()                  t1.wait()
  t2.signal()                t1.signal()
  p_2                        q_2
```

Произволна инструкция `q_2` ще се изпълни, след като изпълняващото я копие на процеса `Q` премине бариерата `t1.wait()`. Бариерата ще се отпуши след изпълнението от поне едно копие на `P` на ред `t1.signal()`, който следва инструкция `p_1`.

Копията на `Q` изпълняват поредица `t1.wait()`, `t1.signal()`. Така семафорът `t1` ще събуди всички приспани други копия на `Q` и ще осигури завършването им.

Аналогично, проиволна инструкция `p_2` ще се изпълни след първото изпълнение на ред `t2.signal()` в някое копие на `Q`, който следва инструкция `q_1`.

Забележка: Някои студенти забелязаха, че може да настъпи препълване на брояча на някой семафор. Това зависи от реализацията на семафора, особено ако брояча се пази в 16 или 32-битово цяло число. Те предложиха по-фино решение, за което даваме до 8 точки бонус:

Задача 1. прецизно решение За двете искани в условието синхронизации използваме два брояча, два семафора – `t1` и `t2` и мутекс `m`, инициализираме ги така:

```
semaphore t1,t2,m
t1.init(0)
t2.init(0)
m.init(1)
int c1=0, c2=0
```

Добавяме в кода на процесите `P` и `Q` синхронизиращи инструкции:

```
process P                  process Q
  p_1                        q_1
  m.wait()                   m.wait()
  if c1=0                    if c2=0
     c1=1                       c2=1
     t1.signal()                t2.signal()
  m.signal()                 m.signal()
  t2.wait()                  t1.wait()
  t2.signal()                t1.signal()
  p_2                        q_2
```

В това решение след изпълнението на `p_1` семафорът `t1` се вдига само веднъж, при първото преминаване през тази критична секция.

Охраняваме броячите с мутекс, но дори да не го направим, броят на паралелно работещи копия, които ще вдигнат семафора е малък и няма да препълни брояча на съответния семафор.

Задача 2. Основните комуникационни канали в ОС Linux са тръба (pipe), именувана тръба (fifo), връзка процес-файл и конекция (изградена с механизма socket).

Всички, освен обикновената тръба, използват пространството на имената.

Именуваната тръба се ползва рядко, ако не я споменете, не губите точки.

Операциите за ползване на канала са общи – read(...), write(...), close().

Специфични са извикванията за изграждане на различните видове канали.