Архитектура операционной системы

Notes on synchronization

Race conditions

Software defect where result is depended on:
  - timings
  - uncontrollable events (like scheduling)
- Types of RC
  - Static
  - Dynamic
  - Essential

Critical section

command sequence to access shared data;

- Correctness if:
  - mutual exclusion
  - Progress
    - at least one process can enter if empty
  - Bounded waiting
    - No process wait indefinitely

High level primitives

- Mutex
- Semaphore
- Critical section
- Conditional variables

Linux sync primitives

- Low level
  - Memory barrier
  - Atomic operations
  - Interrupt synchronization
  - Spin locks
- High level
  - Completion
  - Mutex
  - Semaphore
  - Futex

Deadlock conditions

- Mutual exclusion
- Resource waiting
- No resource relocations
- Circle waiting

Deadlock Prevention

- Eliminate one of 4 conditions
  - Mutex
  - Hold and wait
    - May lead to low resource utilization.
    - Starvation is a problem.
    - If it needs additional resources, it releases all of the currently held resources and then requests all of those it needs
  - No Preemption
  - Circular wait

Approaches

- Lock reordering
- Lock manager
- Waiting graph analysis
- Lock-free algorithms

Classical Problems

- Sleeping barber
- Dining Philosophers
- Producer / Consumer

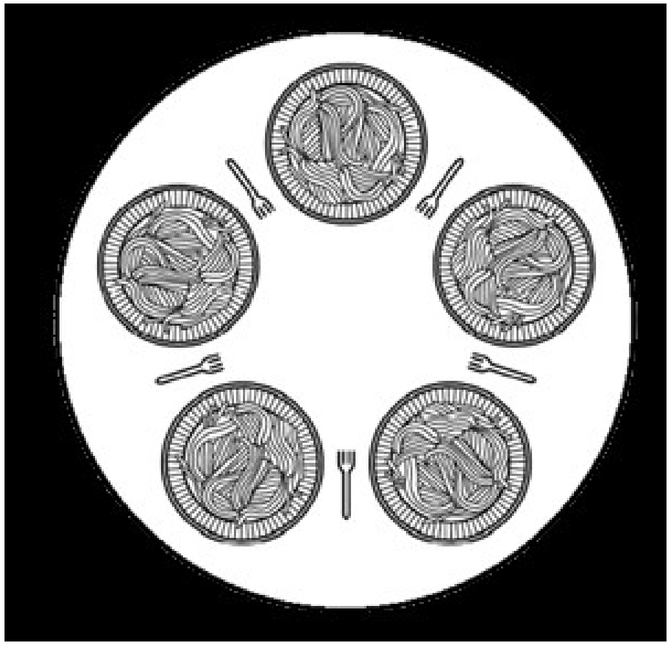
Sleeping barber

The task is modeling queuing system.

- Solution – mutex
- Discussion
  - more than 2 barbers?
- timings

Dining Philosophers

- Problems
  - Starvation
  - Deadlocks & Livelocks
- Solutions
  - Waiter
  - Resource hierarchy
  - Monitors



Producer / consumer problems

The task is to provide synchronized access to shared memory for writers and readers

- Solutions
  - Reader priority
  - Writer priority
  - Fair play

Problems with locking

- Deadlock, livelock
- Priority Inversion
- Convoying

Practice

select

```
int select(int nfds, fd_set *readfds, fd_set *writefds,
           fd_set *exceptfds, struct timeval *timeout);

void FD_CLR(int fd, fd_set *set);
int  FD_ISSET(int fd, fd_set *set);
void FD_SET(int fd, fd_set *set);
void FD_ZERO(fd_set *set);
```

select ex.

```
int
main(void)
{
    fd_set rfds;
    struct timeval tv;
    int retval;

    /* Watch stdin (fd 0) to see when it has input. */
    FD_ZERO(&rfds);
    FD_SET(0, &rfds);

    /* Wait up to five seconds. */
    tv.tv_sec = 5;
    tv.tv_usec = 0;

    retval = select(1, &rfds, NULL, NULL, &tv);
    /* Don't rely on the value of tv now! */

    if (retval == -1)
        perror("select()");
    else if (retval)
        printf("Data is available now.\n");
        /* FD_ISSET(0, &rfds) will be true. */
    else
        printf("No data within five seconds.\n");

    exit(EXIT_SUCCESS);
}
```

Инициализация

Ожидание

Проверка

Домашнее задание

- Курс: https://stepik.org/course/1780
  - 4. Средства синхронизации потоков
- Контейнерная вируализация в Linux