ЛЕКЦ 2,3,4

Үйлдлийн системийн онол

Хичээл заах багш: доктор, дэд профессор
О. Бат-Энх

Үйлдлийн системийн онол

Бүлэг 2
Процессууд ба хуулбар процессууд

Процессын хэрэгжүүлэлт (3)

1. Hardware stacks program counter, etc.
2. Hardware loads new program counter from interrupt vector.
3. Assembly language procedure saves registers.
4. Assembly language procedure sets up new stack.
5. C interrupt service runs (typically reads and buffers input).
6. Scheduler decides which process is to run next.
7. C procedure returns to the assembly code.
8. Assembly language procedure starts up new current process.

Figure 2-5. Skeleton of what the lowest level of the operating system does when an interrupt occurs.

Олон программчлалын загвар

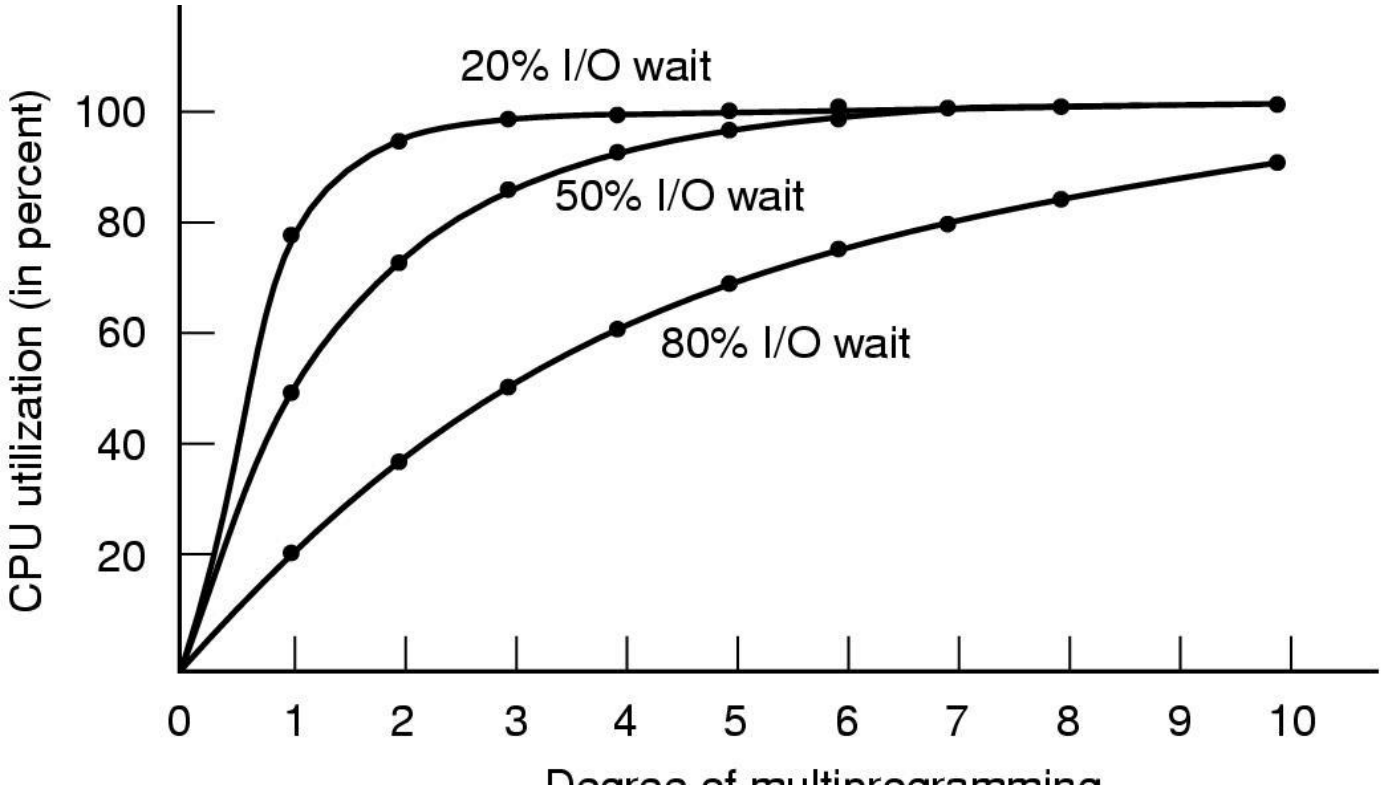
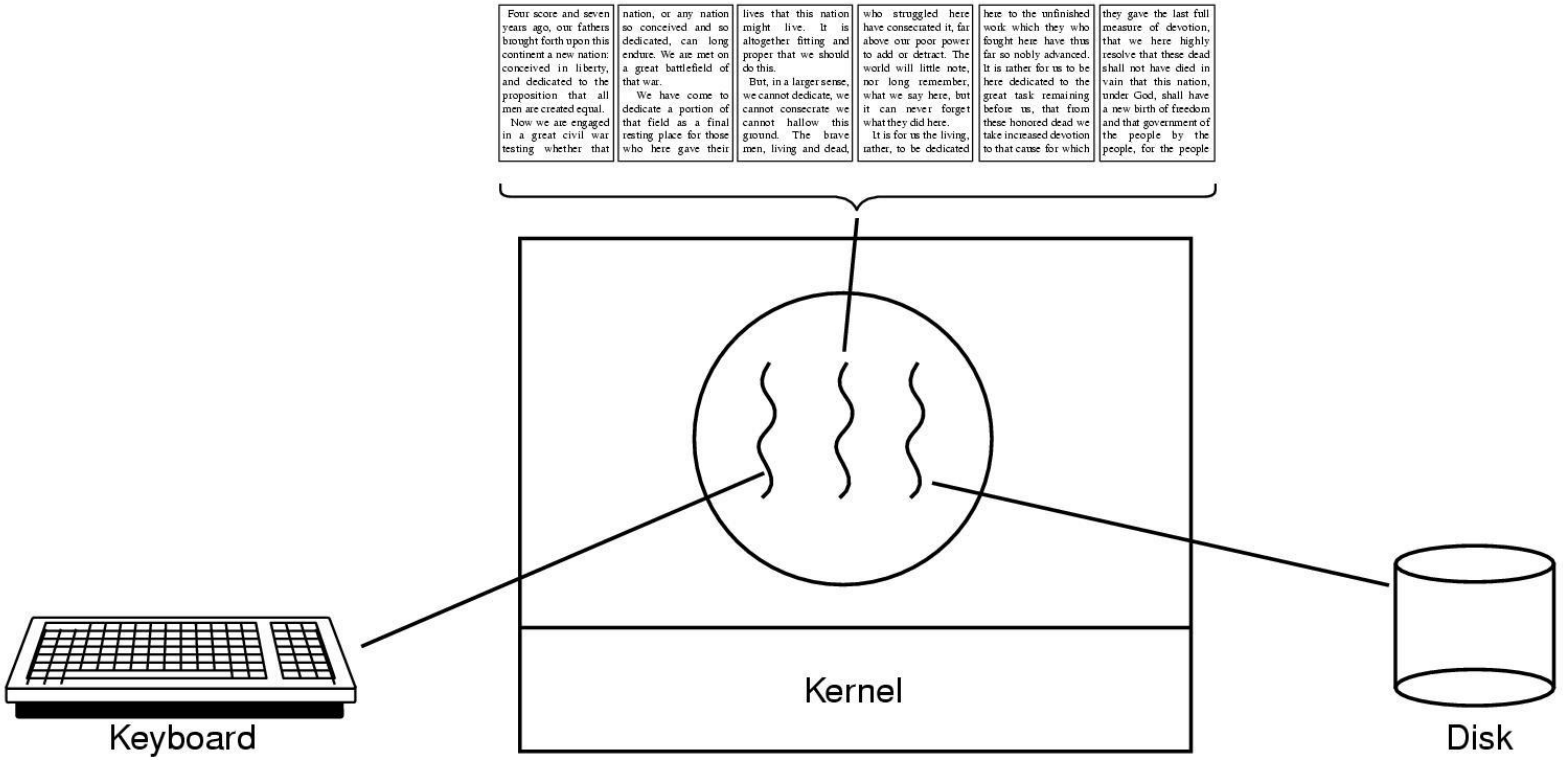


Figure 2-6. CPU utilization as a function of the number of processes in memory.

Салбар процессын хэрэглээ (1)



Зураг 2-7. 3 салбар процесстой word processor.

Салбар процессын хэрэглээ(2)

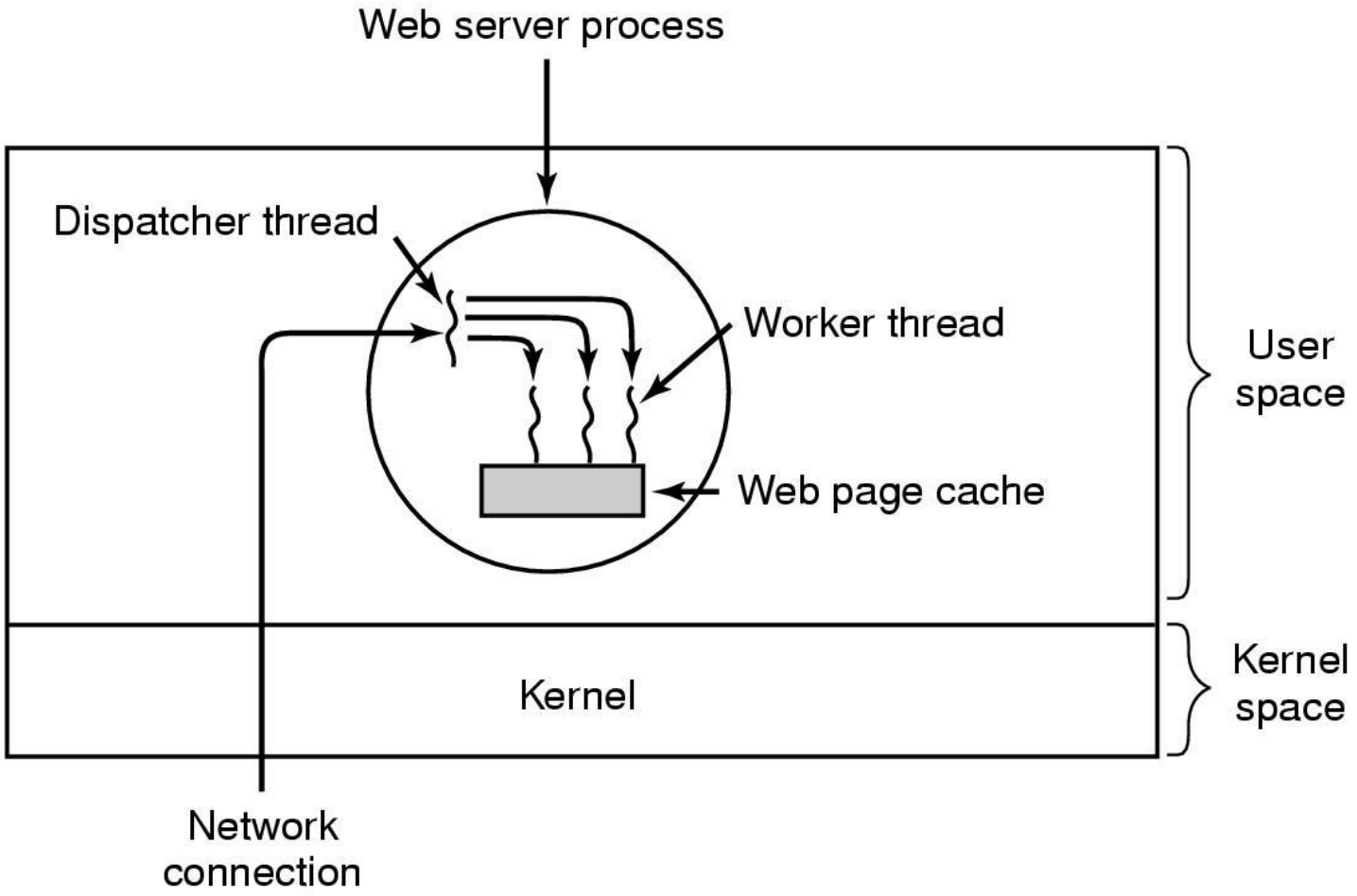


Figure 2-8. A multithreaded Web server.

Салбар процессын хэрэглээ (3)

```
while (TRUE) {
    get_next_request(&buf);
    handoff_work(&buf);
}
```

(a)

```
while (TRUE) {
    wait_for_work(&buf)
    look_for_page_in_cache(&buf, &page);
    if (page_not_in_cache(&page))
        read_page_from_disk(&buf, &page);
    return_page(&page);
}
```

(b)

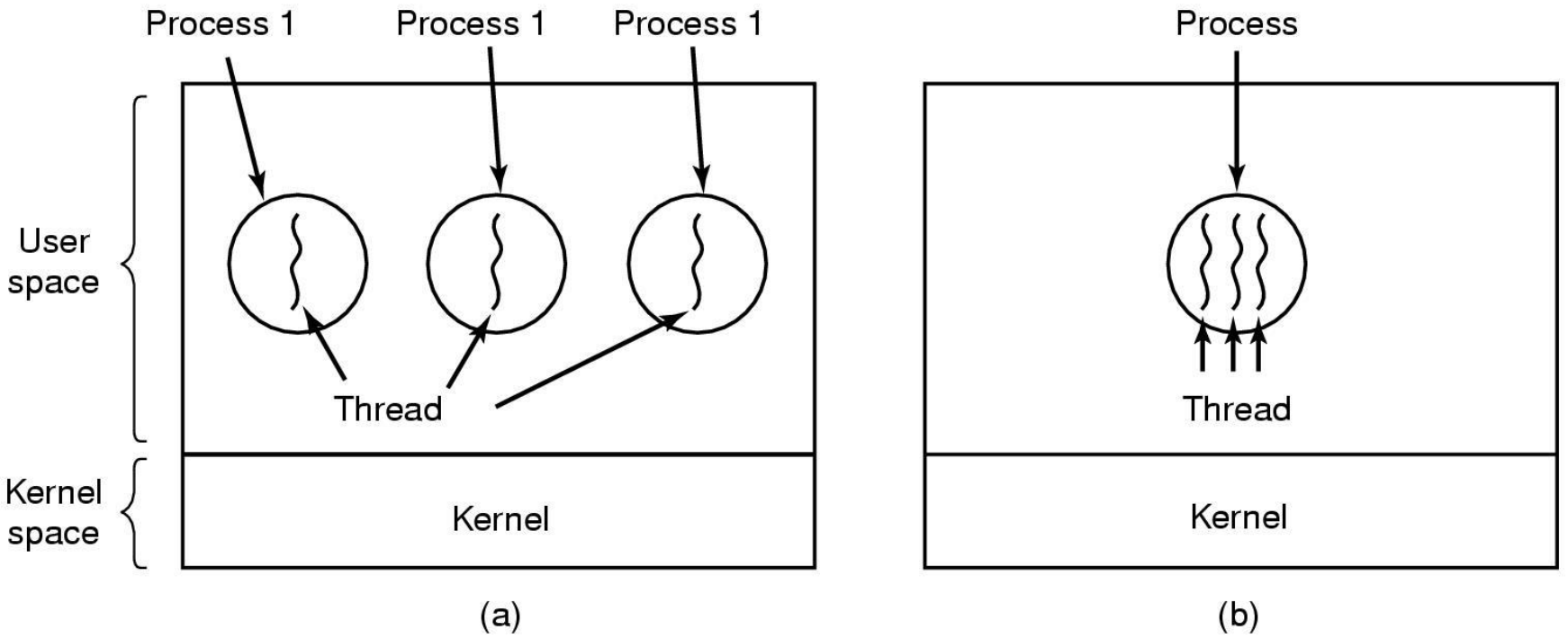
Figure 2-9. A rough outline of the code for Fig. 2-8. (a) Dispatcher thread. (b) Worker thread.

Салбар процессын хэрэглээ (4)

Model	Characteristics
Threads	Parallelism, blocking system calls
Single-threaded process	No parallelism, blocking system calls
Finite-state machine	Parallelism, nonblocking system calls, interrupts

Figure 2-10. Three ways to construct a server.

Салбар процессын загвар (1)



Зураг 2-11. (a) Тус бүрдээ нэг салбар процесстой 3 процесс. (b) 3 салбар процесстой 1 процесс.

Салбар процессын загвар (2)

Per process items	Per thread items
Address space	Program counter
Global variables	Registers
Open files	Stack
Child processes	State
Pending alarms	
Signals and signal handlers	
Accounting information	

Figure 2-12. The first column lists some items shared by all threads in a process. The second one lists some items private to each thread.

Салбар процессын загвар (3)

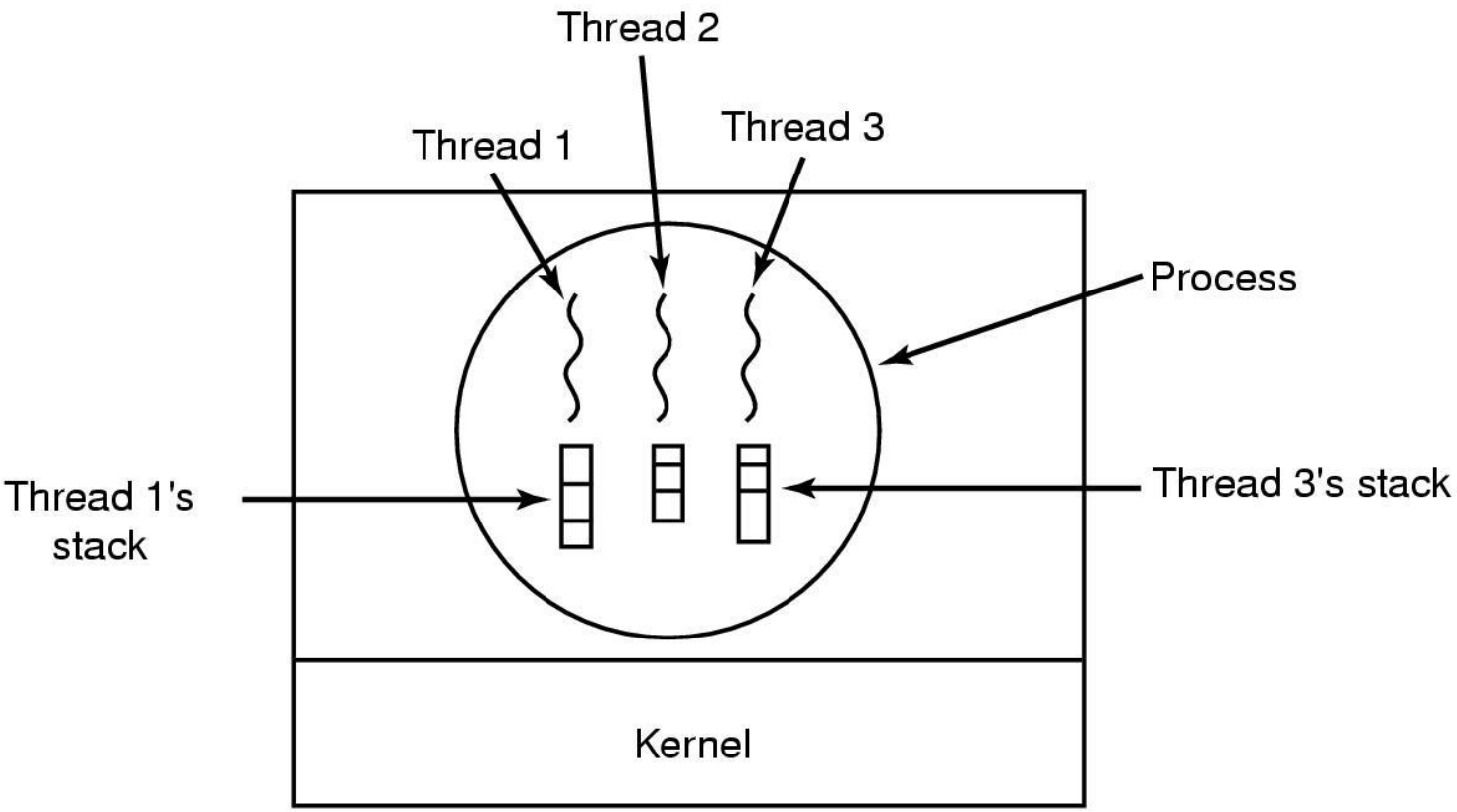
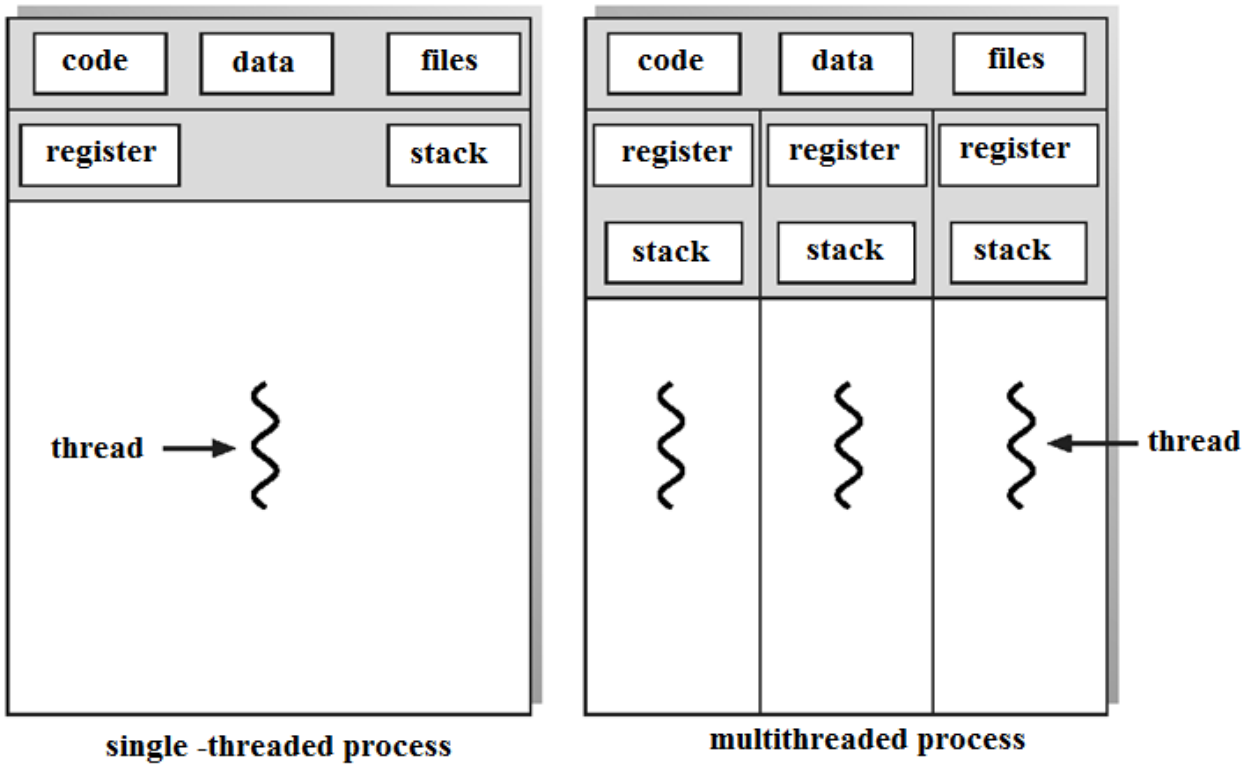


Figure 2-13. Each thread has its own stack.

Нэг ба олон салбар процессуудтай процессууд.



POSIX салбар процессууд (1)

Thread call	Description
Pthread_create	Create a new thread
Pthread_exit	Terminate the calling thread
Pthread_join	Wait for a specific thread to exit
Pthread_yield	Release the CPU to let another thread run
Pthread_attr_init	Create and initialize a thread's attribute structure
Pthread_attr_destroy	Remove a thread's attribute structure

Figure 2-14. Some of the Pthreads function calls.

POSIX салбар процессууд (2)

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define NUMBER_OF_THREADS     10

void *print_hello_world(void *tid)
{
        /* This function prints the thread's identifier and then exits. */
        printf("Hello World. Greetings from thread %d0, tid);
        pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
        /* The main program creates 10 threads and then exits. */
        pthread_t threads[NUMBER_OF_THREADS];
        int status, i;

        for(i=0; i < NUMBER_OF_THREADS; i++) {
                printf("Main here. Creating thread %d0, i);
                status = pthread_create(&threads[i], NULL, print_hello_world, (void *)i);

                if (status != 0) {
                        printf("Oops. pthread_create returned error code %d0, status);
                        exit(-1);
                }
        }
        exit(NULL);
}
```

Figure 2-15. An example program using threads.

Хэрэглэгчийн орон зай дахь салбар процессуудын хэрэгжилт

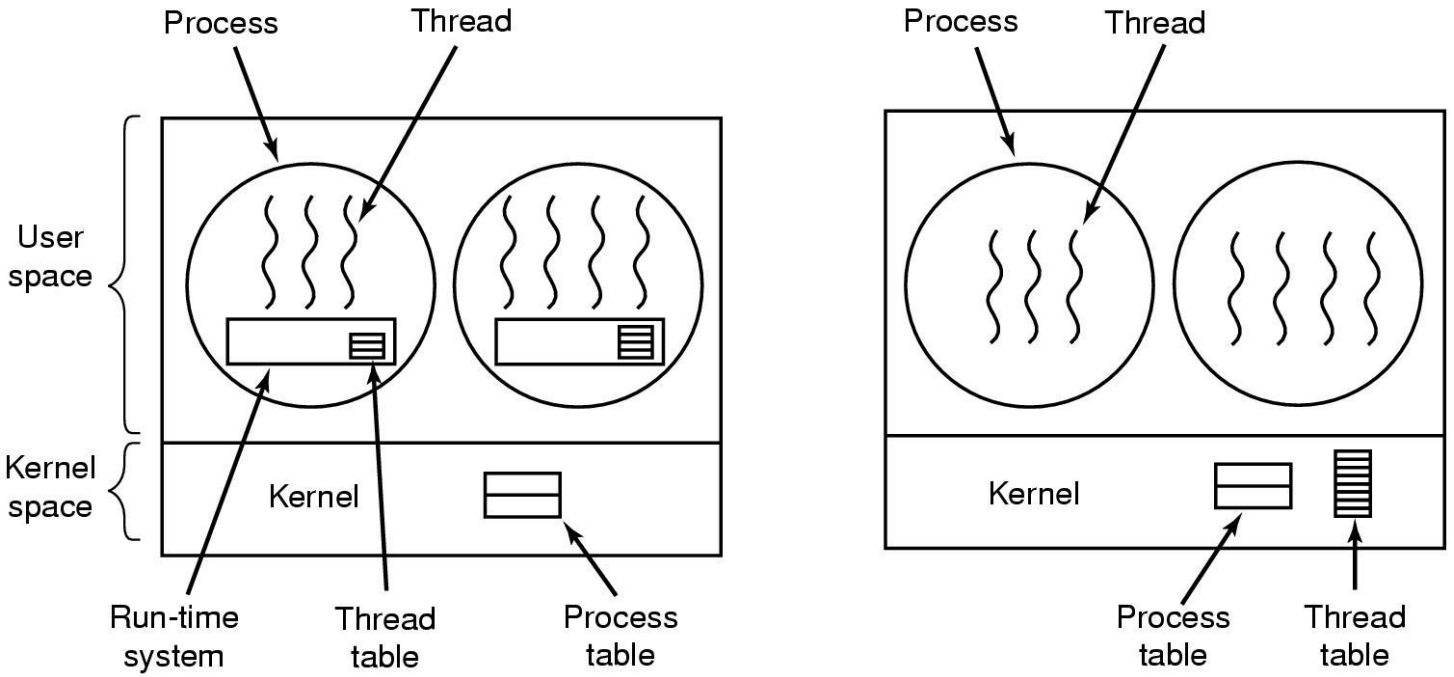


Figure 2-16. (a) A user-level threads package. (b) A threads package managed by the kernel.

Холимог хэрэгжилт

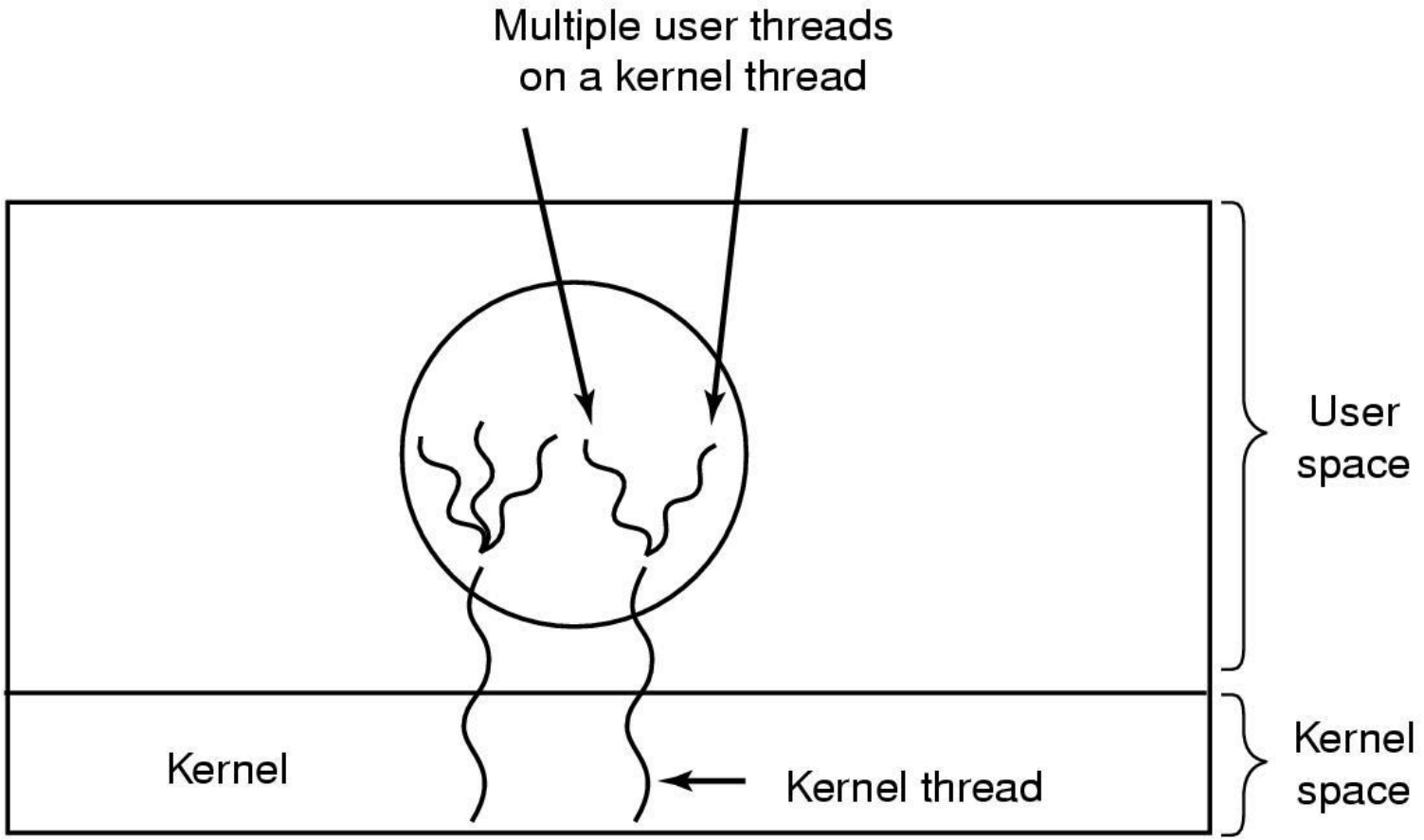


Figure 2-17. Multiplexing user-level threads onto kernel-level threads.

Pop-Up салбар процессууд

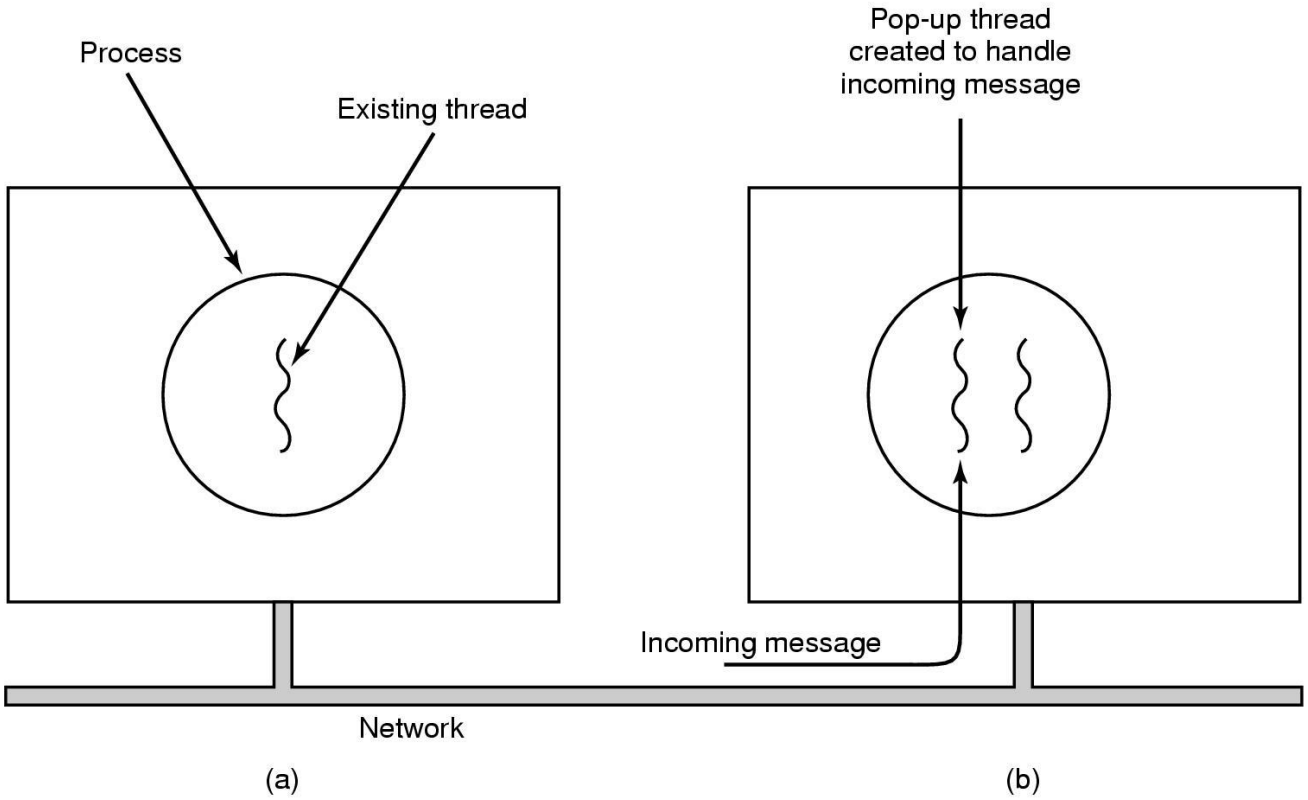


Figure 2-18. Creation of a new thread when a message arrives.
(a) Before the message arrives.
(b) After the message arrives.

Making Single-Threaded Code Multithreaded (1)

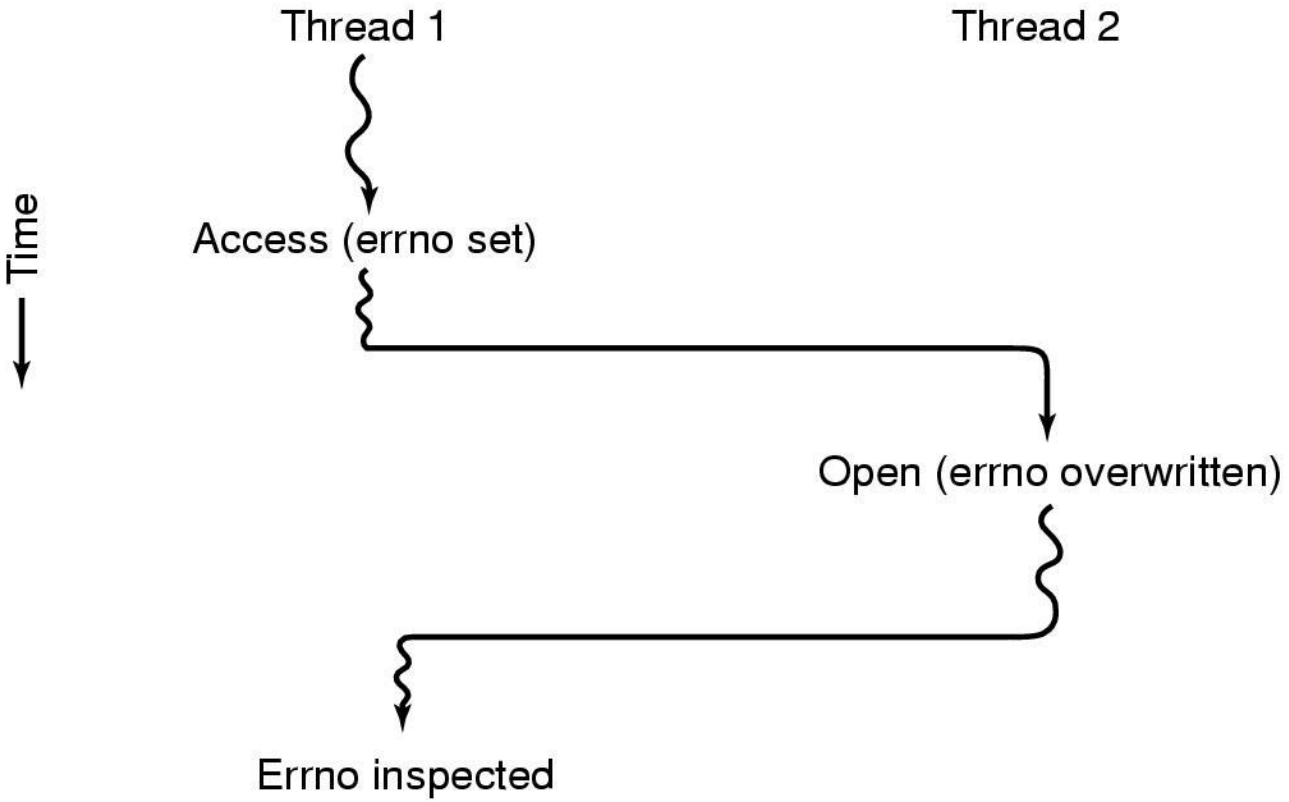


Figure 2-19. Conflicts between threads over the use of a global variable.

Making Single-Threaded Code Multithreaded (2)

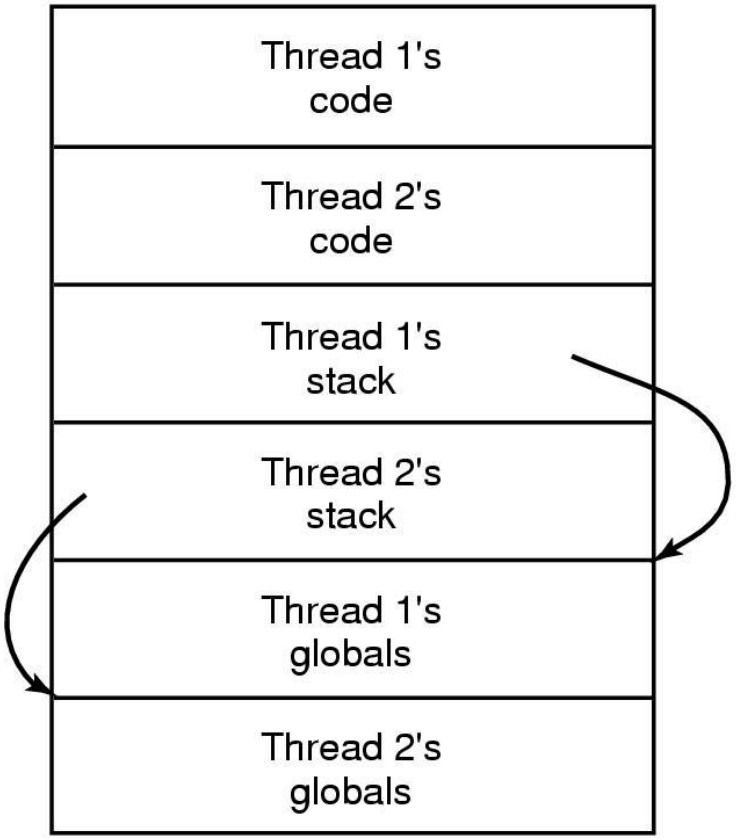


Figure 2-20. Threads can have private global variables.

Уралдах нөхцөл

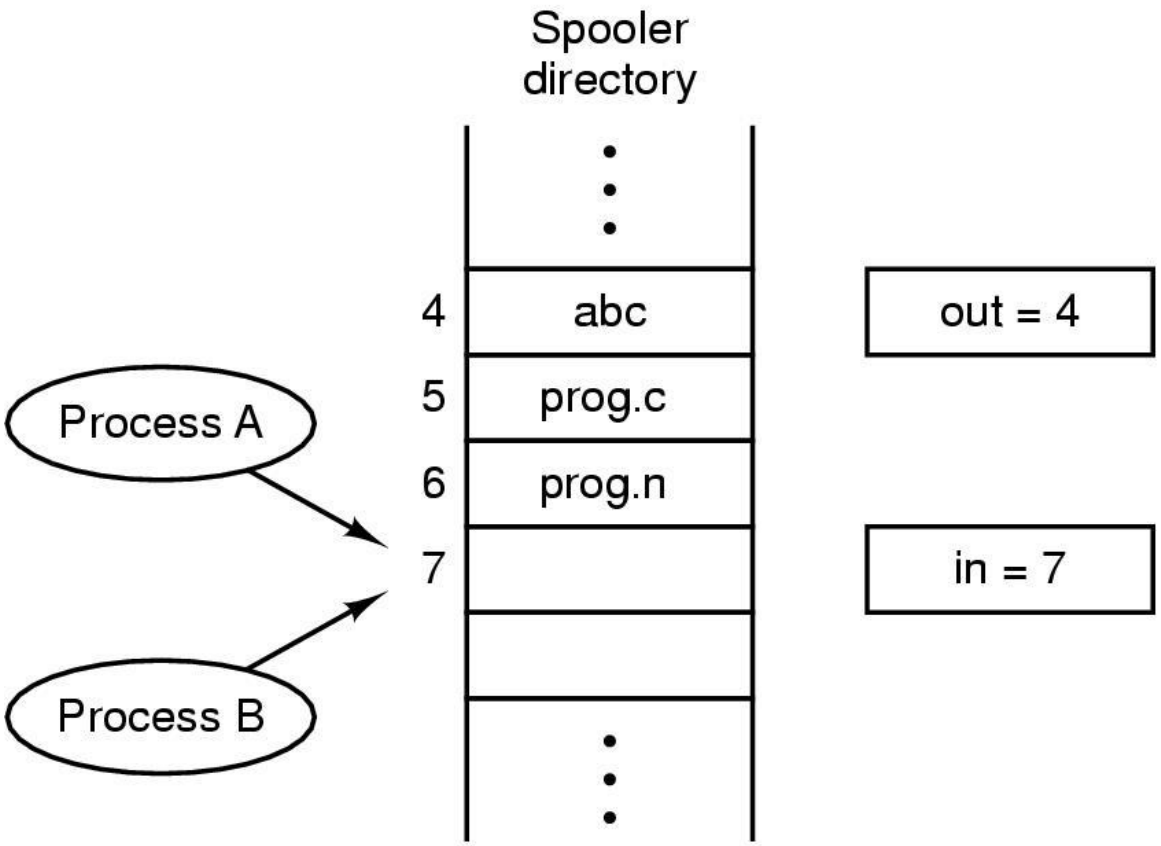


Figure 2-21. Two processes want to access shared memory at the same time.

Сэжигтэй муж (2)

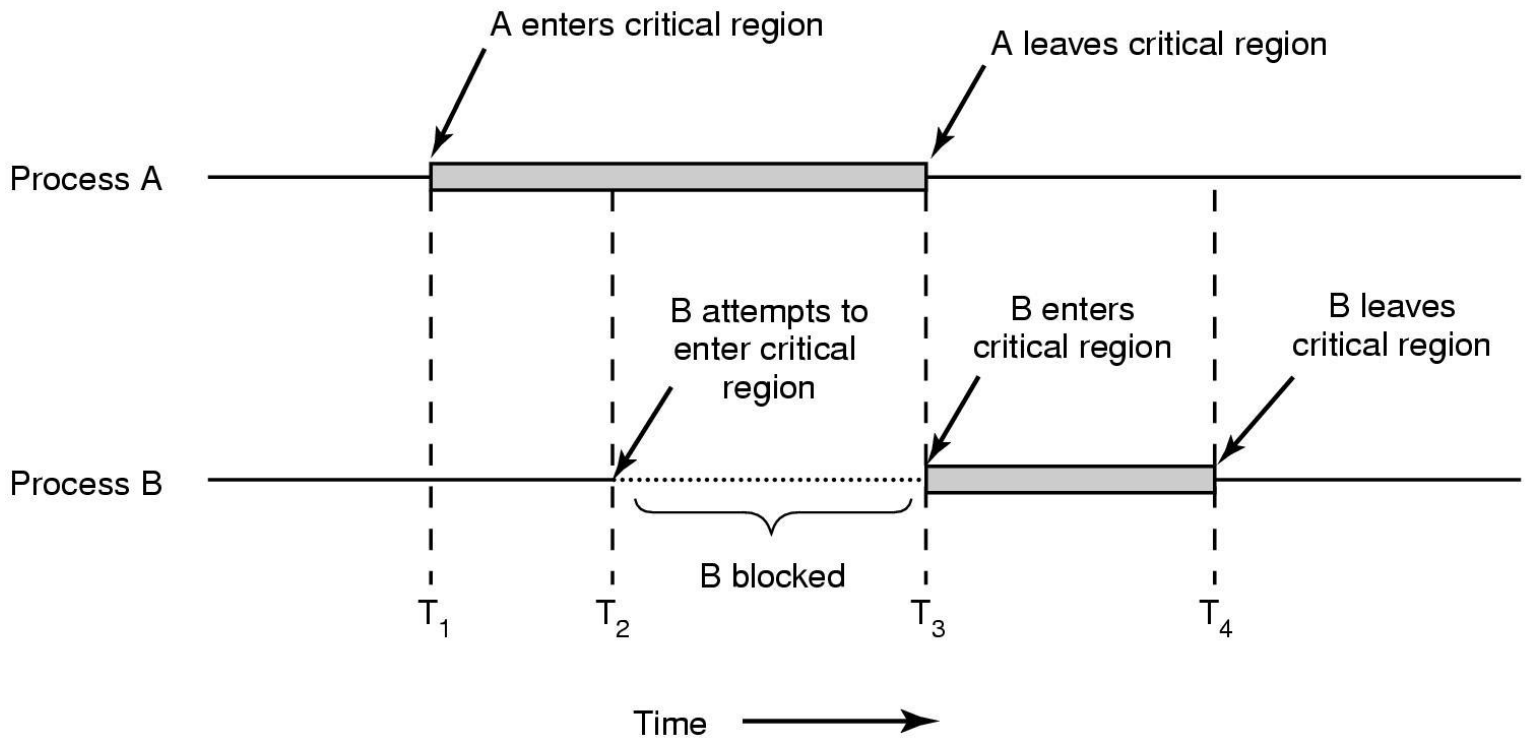


Figure 2-22. Mutual exclusion using critical regions.

Төлөвлөлт – Процессын хэв маяг

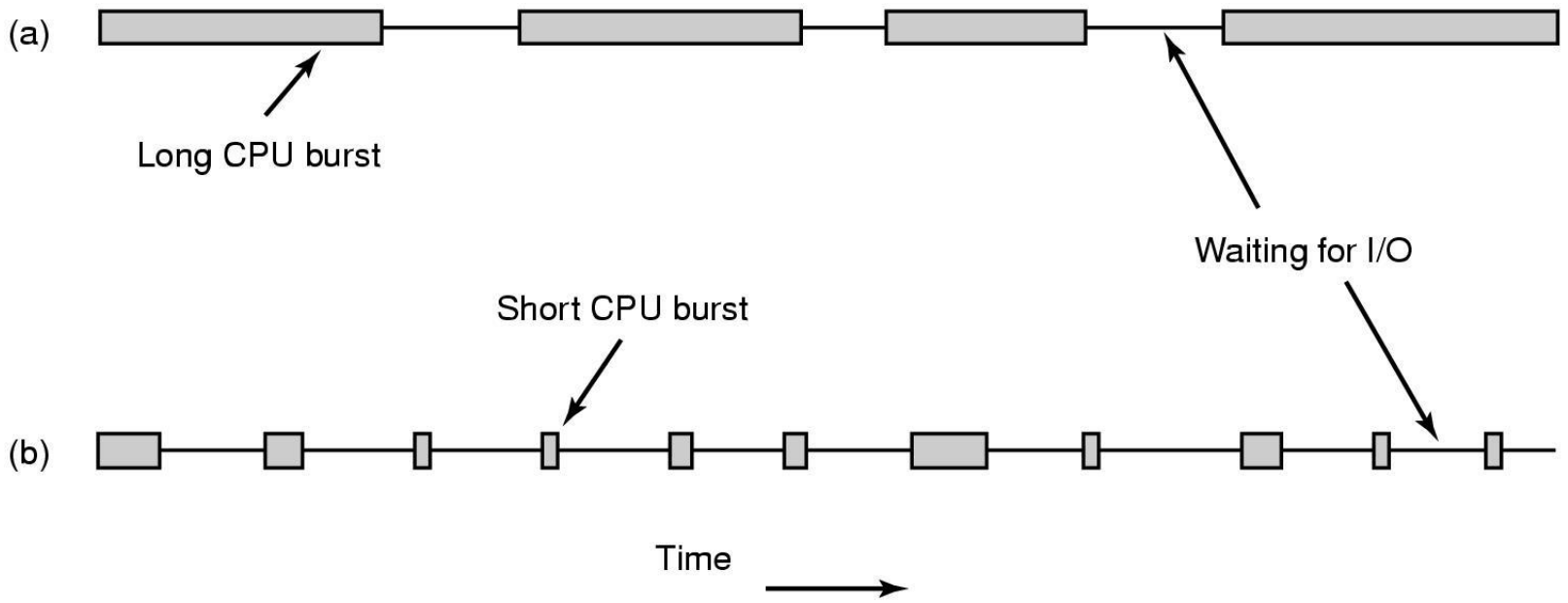


Figure 2-38. Bursts of CPU usage alternate with periods of waiting for I/O. (a) A CPU-bound process. (b) An I/O-bound process.

Төлөвлөлтийн алгоритмууд

- Batch
- Interactive
- Real time

Төлөвлөлтийн алгоритмтын зорилго

All systems

- Fairness - giving each process a fair share of the CPU
- Policy enforcement - seeing that stated policy is carried out
- Balance - keeping all parts of the system busy

Batch systems

- Throughput - maximize jobs per hour
- Turnaround time - minimize time between submission and termination
- CPU utilization - keep the CPU busy all the time

Interactive systems

- Response time - respond to requests quickly
- Proportionality - meet users' expectations

Real-time systems

- Meeting deadlines - avoid losing data
- Predictability - avoid quality degradation in multimedia systems

Figure 2-39. Some goals of the scheduling algorithm under different circumstances.

Batch систем дэх төлөвлөлт

- First-come first-served
- Shortest job first
- Shortest remaining Time next

Process	Service time T_s	Turnaround time T_r	Waiting time T_w
A	10	10	0
B	1	11	10
C	3	14	11
D	4	18	14
Average		13.25	8.75
$T_r / T_w = 1.51$		$T_r / T_s = 5.29$	

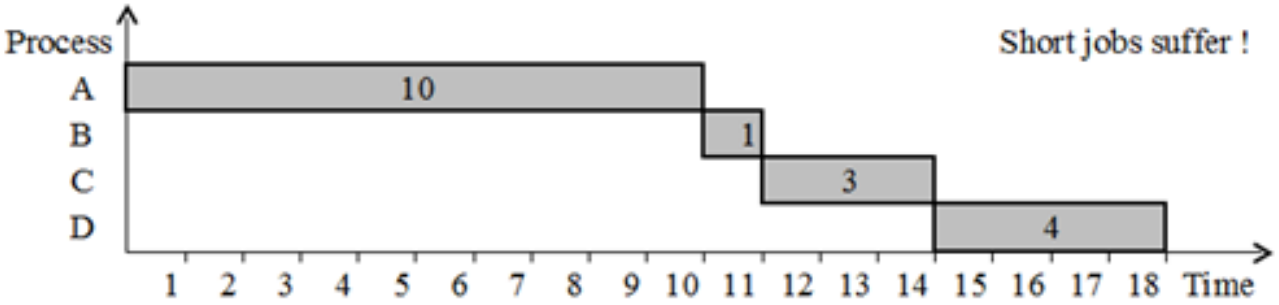


Figure 3: An example to First-Come First-Served.

Shortest Job First

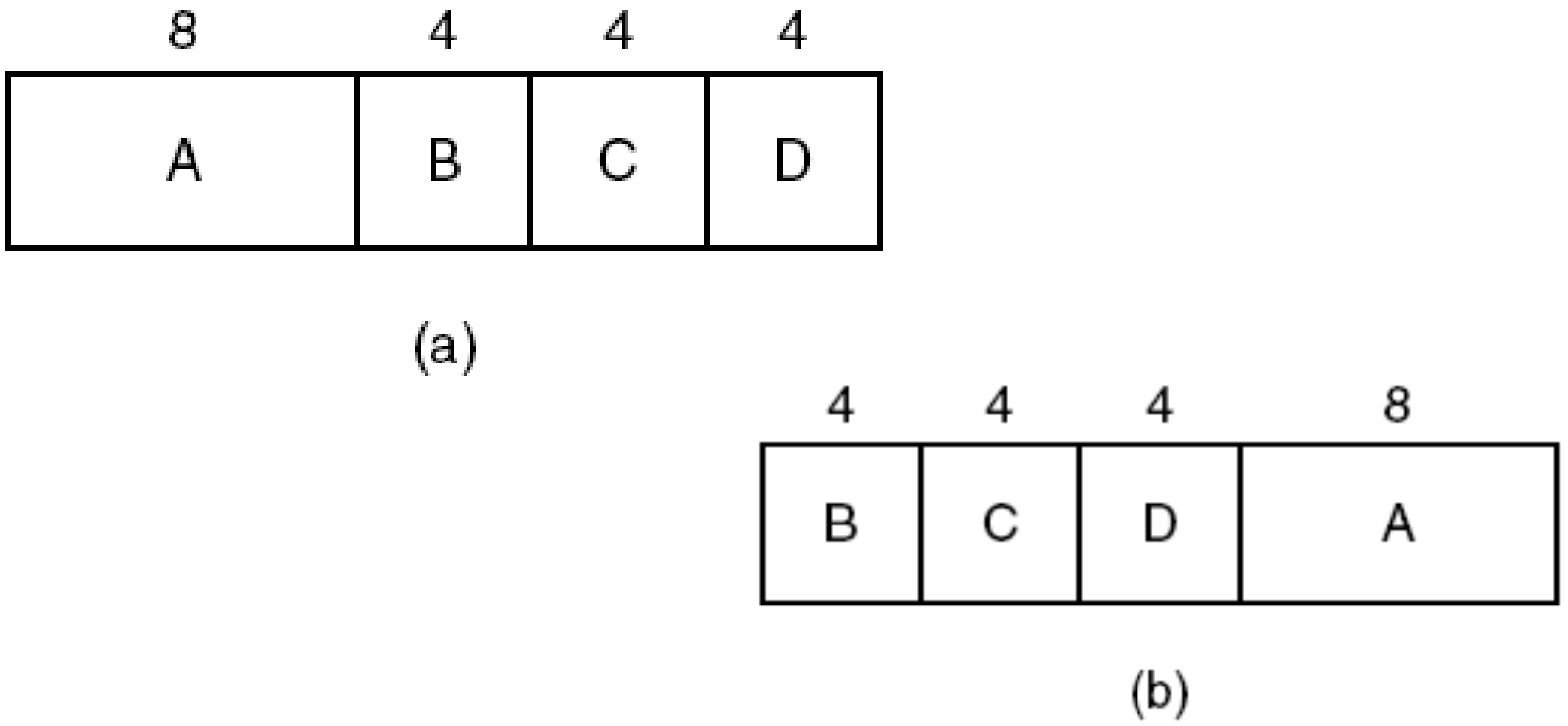


Figure 2-40. An example of shortest job first scheduling. (a) Running four jobs in the original order. (b) Running them in shortest job first order.

Process	Service time T_s	Turnaround time T_r	Waiting time T_w
A	10	18	8
B	1	1	0
C	3	4	1
D	4	8	4
Average			
$T_r / T_w = 2.38$		$T_r / T_s = 1.53$	

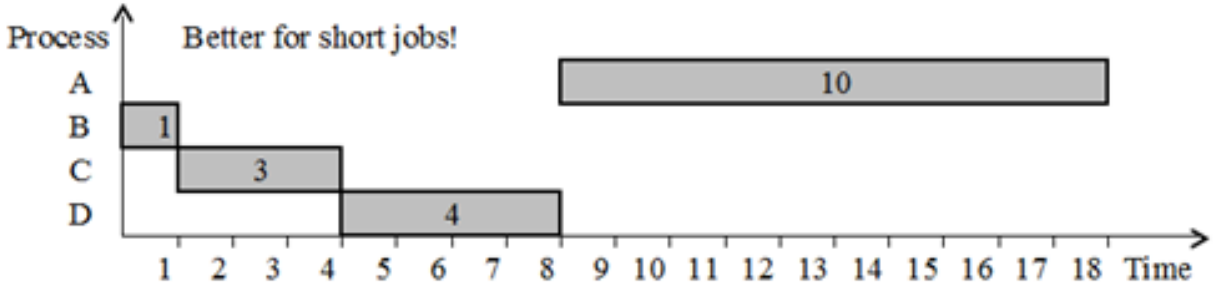


Figure 5: An example to Shortest Job First.

Scheduling in Interactive Systems

- Round-robin scheduling
- Priority scheduling
- Multiple queues
- Shortest process next
- Guaranteed scheduling
- Lottery scheduling
- Fair-share scheduling

Round-Robin Scheduling

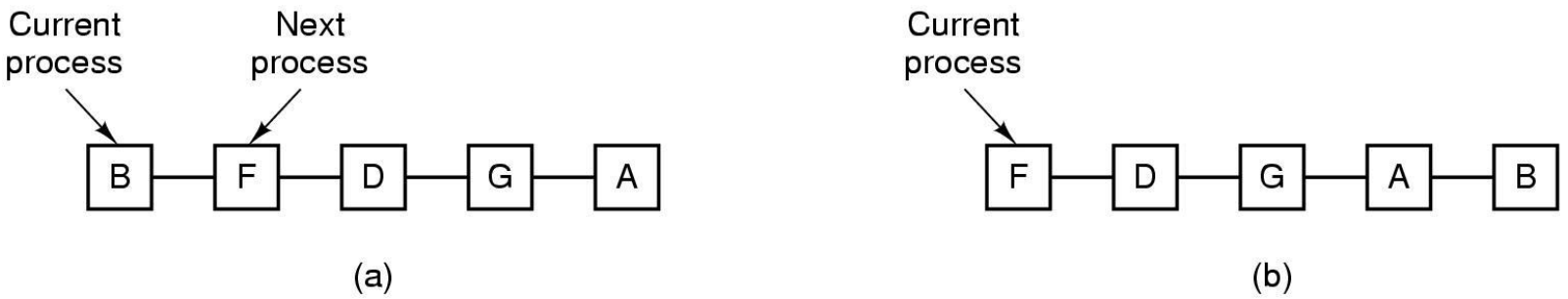


Figure 2-41. Round-robin scheduling.
(a) The list of runnable processes. (b) The list of runnable processes after B uses up its quantum.

Process	Service time T_s	Turnaround time T_r	Waiting time T_w
A	10	18	8
B	1	2	1
C	3	9	6
D	4	12	8
Average		10.3	5.75
$T_r / T_w = 1.78$		$T_r / T_s = 2.45$	

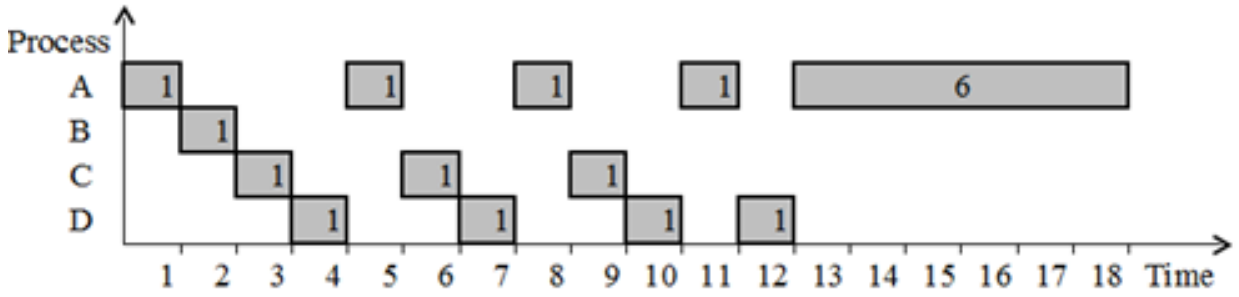


Figure 8: An example to Round Robin.

Priority Scheduling

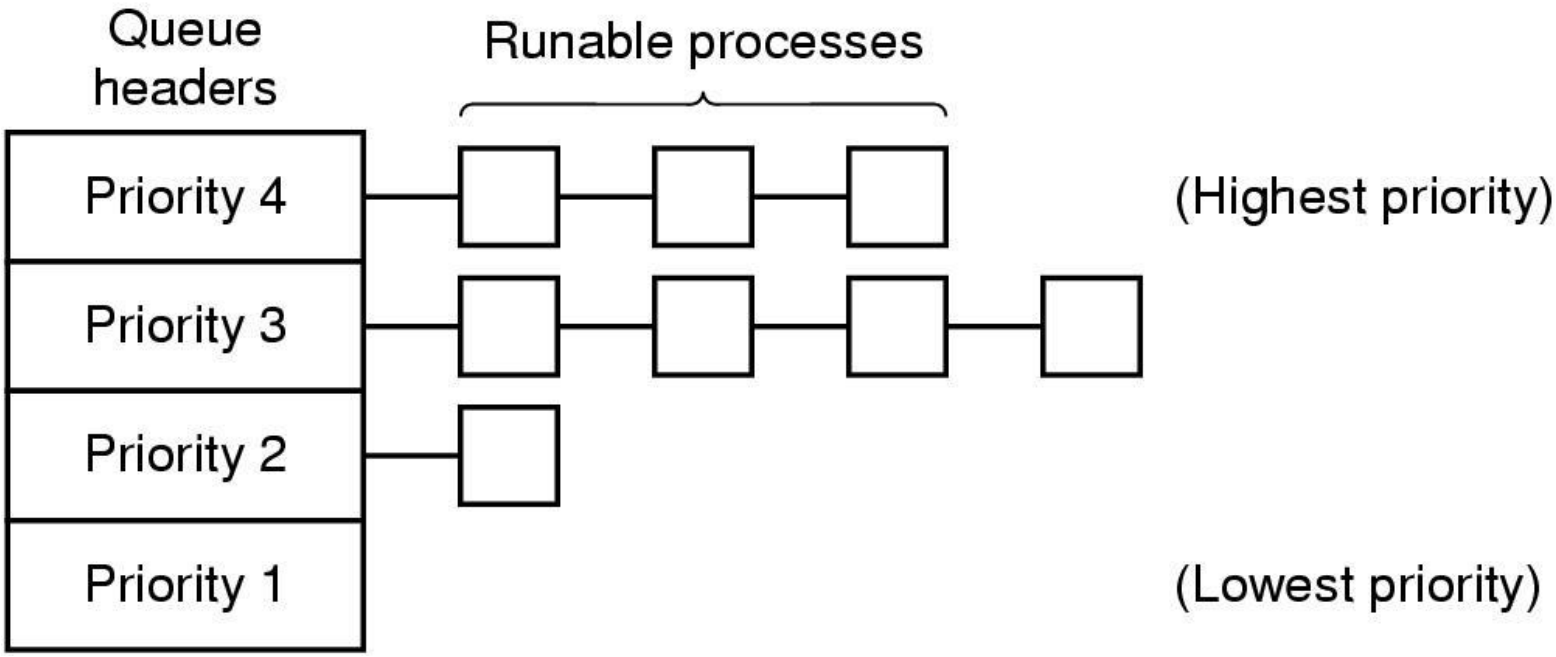


Figure 2-42. A scheduling algorithm with four priority classes.

Process	Priority	Service time T_s	Turnaround time T_r	Waiting time T_w
A	4	10	18	8
B	3	1	8	7
C	2	3	7	4
D	1	4	4	0
Average		4.5	9.25	4.75
$T_r / T_w = 1.95$			$T_r / T_s = 3.28$	

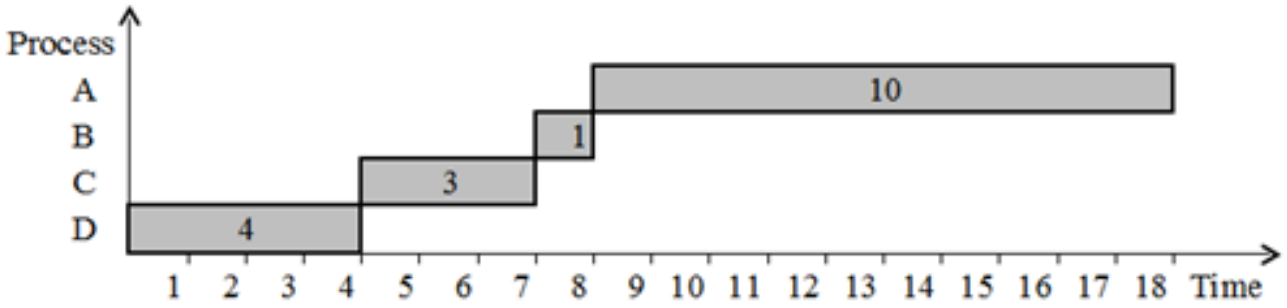


Figure 10: An example to Priority-based Scheduling.