



API

system call

standard library

CPU

resource manager

CPU  
manage

resource

\$~#

5BG

2.1

Spin()

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/time.h>
4  #include <assert.h>
5  #include "common.h"
6
7  int
8  main(int argc, char *argv[])
9  {
10     if (argc != 2) {
11         fprintf(stderr, "usage: cpu <string>\n");
12         exit(1);
13     }
14     char *str = argv[1];
15     while (1) {
16         Spin(1);
17         printf("%s\n", str);
18     }
19     return 0;
20 }

```

2.1

cpu.c

cpu.c

CPU

prompt&gt; gcc -o cpu cpu.c -Wall

prompt&gt; ./cpu "A"

A

A

```
A
A
C
prompt>
```

Control-c

UNIX

A

## 2.2

```
prompt> ./cpu A & ; ./cpu B & ; ./cpu C & ; ./cpu D &
[1] 7353
[2] 7354
[3] 7355
[4] 7356
A
B
D
C
A
B
D
C
A
C
B
D
...
```

2.2

4

illusion

CPU

CPU

CPU virtualizing the CPU

CPU

API

API

&amp;

(

tcsh shell

tcsh

shell

bash

policy

mechanism

resource manager

\$Z\$

physical memory  
addressread  
write      update

malloc()

2.3

```

1  #include <unistd.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include "common.h"
5
6  int
7  main(int argc, char *argv[])
8  {
9      int *p = malloc(sizeof(int));           // a1
10     assert(p != NULL);
11     printf("(%d) memory address of p: %08x\n",
12           getpid(), (unsigned) p);          // a2
13     *p = 0;                                  // a3
14     while (1) {
15         Spin(1);
16         *p = *p + 1;
17         printf("(%d) p: %d\n", getpid(), *p); // a4
18     }
19     return 0;
20 }
```

2.3

mem.c

```

prompt> ./mem
(2134) memory address of p: 00200000
(2134) p: 1
(2134) p: 2
(2134) p: 3
(2134) p: 4
(2134) p: 5
```

8

0 a3 a1 a2

p

PID PID

00200000

2.4

00200000

00200000

```
prompt> ./mem &i ./mem &
[1] 24113
[2] 24114
(24113) memory address of p: 00200000
(24114) memory address of p: 00200000
(24113) p: 1
(24114) p: 1
(24114) p: 2
(24113) p: 2
(24113) p: 3
(24114) p: 3
(24113) p: 4
(24114) p: 4
...
```

2.4

virtualizing memory

virtual address space

address space

1

virtualization

\$2%

concurrency

multi-threaded

2.5

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "common.h"
4
5  volatile int counter = 0;
6  int loops;
7
8  void *worker(void *arg) {
9      int i;
10     for (i = 0; i < loops; i++) {
11         counter++;
12     }
13     return NULL;
14 }
15
16 int
17 main(int argc, char *argv[])
18 {
19     if (argc != 2) {
20         fprintf(stderr, "usage: threads <value>\n");
21         exit(1);
22     }
23     loops = atoi(argv[1]);
24     pthread_t p1, p2;
25     printf("Initial value : %d\n", counter);
26
27     Pthread_create(&p1, NULL, worker, NULL);
28     Pthread_create(&p2, NULL, worker, NULL);
29     Pthread_join(p1, NULL);
30     Pthread_join(p2, NULL);
31     printf("Final value      : %d\n", counter);
32     return 0;
33 }

```

2.5

threads.c

Pthread\_create()

thread

worker()

loops

loops

1000

loops

worker

loops

1000

pthread\_create()

pthread\_create()

```

prompt> gcc -o thread thread.c -Wall -pthread
prompt> ./thread 1000
Initial value : 0
Final value   : 2000

```

loops  $N$  2000 1000  
 $2N$   
loops

```

prompt> ./thread 100000
Initial value : 0
Final value   : 143012 // huh??
prompt> ./thread 100000
Initial value : 0
Final value   : 137298 // what the??

```

143012 100000 200000  
loops

atomically  
concurrency 3 3  
2

\$z&

DRAM persistence  
volatile  
persistently  
/ Input/Output I/O  
hard drive Solid-State Drive  
SSD  
file system  
file  
CPU  
share  
Emacs C  
C emacs -nw main.c  
gcc -o main main.c

./main

Emacs

2.6

hello world /tmp/file

```

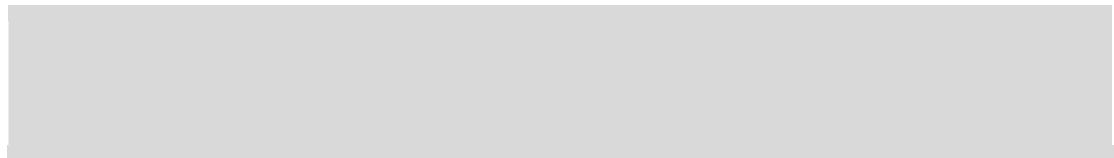
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <assert.h>
4  #include <fcntl.h>
5  #include <sys/types.h>
6
7  int
8  main(int argc, char *argv[])
9  {
10     int fd = open("/tmp/file", O_WRONLY | O_CREAT | O_TRUNC, S_IRWXU);
11     assert(fd > -1);
12     int rc = write(fd, "hello world\n", 13);
13     assert(rc == 13);
14     close(fd);
15     return 0;
16 }

```

2.6

I/O

io.c



3

open()

write()

close()

system call

file system

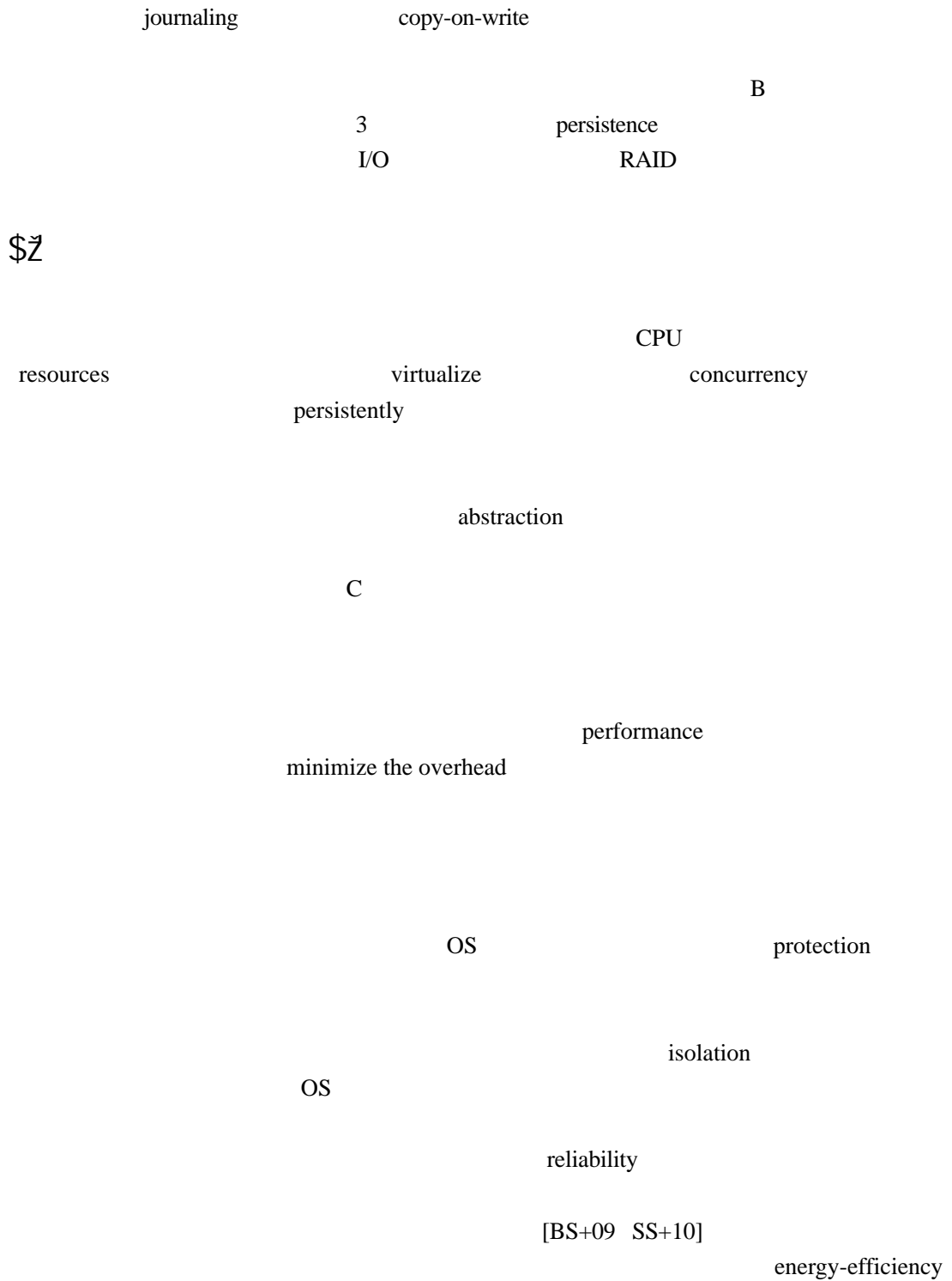
I/O

device driver

OS

standard library





security

mobility

\$Ź

Brinch

Hansen

[BH00]

I/O

OS

API

batch

[BH00]

file system

system call

Atlas

[K+61 L78]

procedure call

hardware privilege level

OS

user mode

---

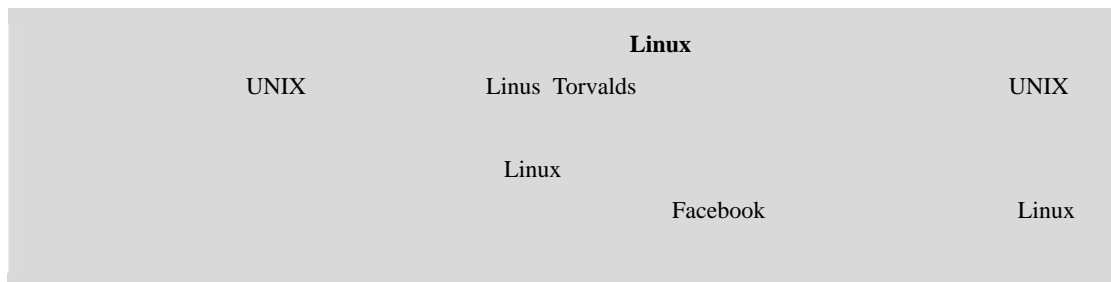
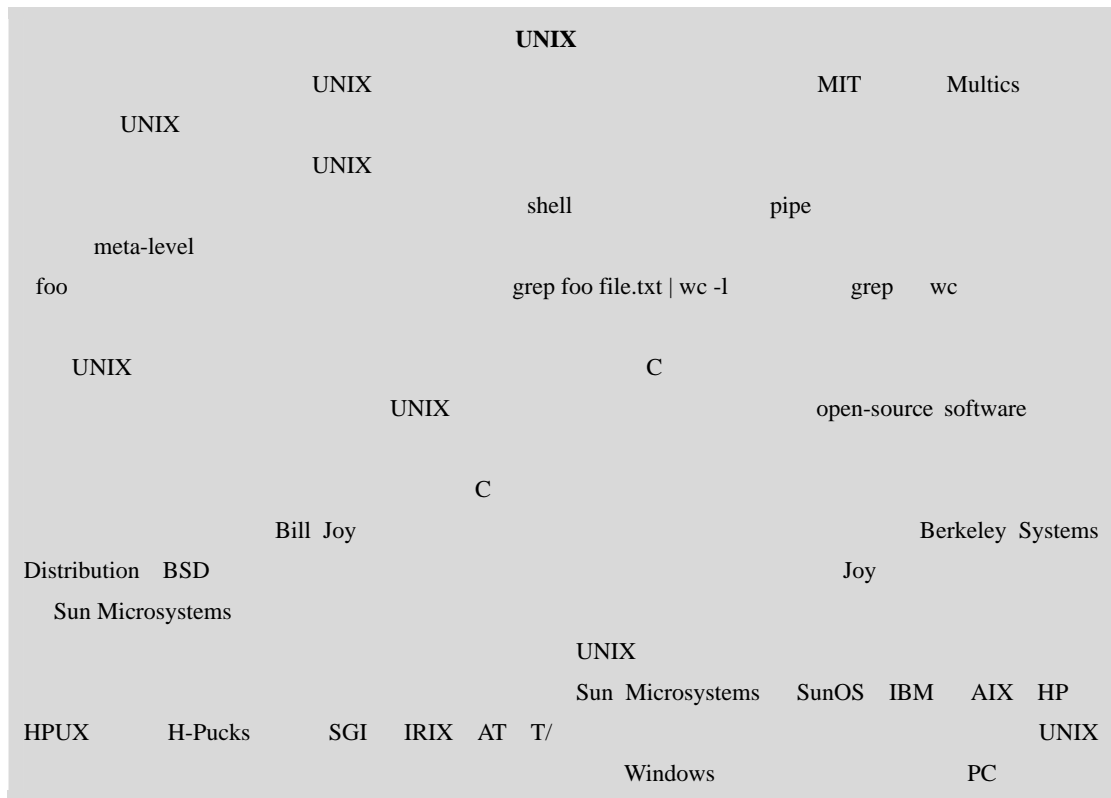
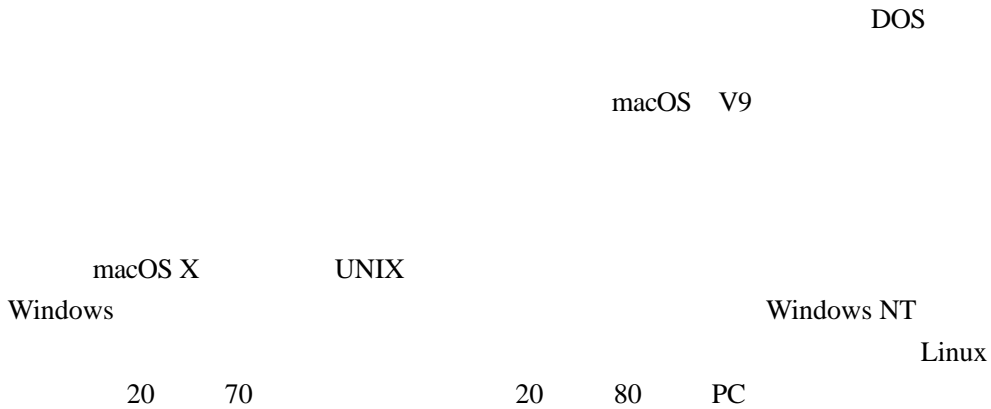
I/O  
 [ trap ]  
 trap handler  
 mode  
 kernel  
 I/O  
 return-from-trap

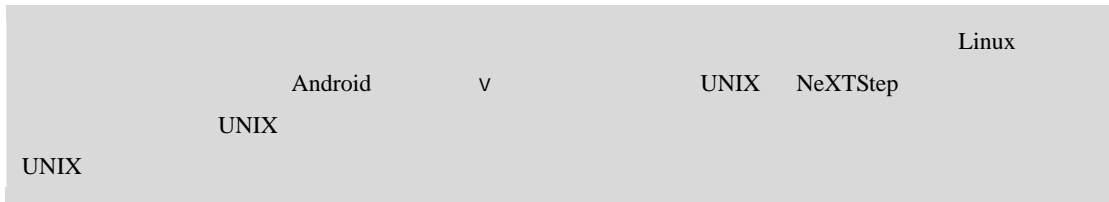
minicomputer  
 DEC PDP  
 multiprogramming  
 CPU I/O I/O  
 CPU CPU  
 I/O  
 memory protection  
 concurrency

UNIX  
 Ken Thompson Dennis Ritchie UNIX  
 Multics [O72] TENEX [B+72] Berkeley [S+68]  
 UNIX

IBM PC Personal Computer PC Apple II

---





\$Z)

CPU

[BS+09] Tolerating File-System Mistakes with EnvyFS

Lakshmi N. Bairavasundaram, Swaminathan Sundararaman, Andrea C. Arpaci-Dusseau, Remzi

H. Arpaci-Dusseau

USENIX ~~09~~, San Diego, CA, June 2009

[BH00] The Evolution of Operating Systems

P. Brinch Hansen

In Classic Operating Systems: From Batch Processing to Distributed Systems Springer-Verlag, New York, 2000

[B+72] TENEX, A Paged Time Sharing System for the PDP-10

Daniel G. Bobrow, Jerry D. Burchfiel, Daniel L. Murphy, Raymond S. Tomlinson CACM, Volume 15, Number 3, March 1972

TENEX

20 70

[B75] The Mythical Man-Month Fred Brooks

Addison-Wesley, 1975

[BOH10] Computer Systems: A Programmer's Perspective Randal E. Bryant and David R. O'Hallaron  
Addison-Wesley, 2010

[K+61] One-Level Storage System

T. Kilburn, D.B.G. Edwards, M.J. Lanigan, F.H. Sumner IRE Transactions on Electronic Computers, April 1962  
Atlas

[L78]

[L78] The Manchester Mark I and Atlas: A Historical Perspective

S.H. Lavington

Communications of the ACM archive Volume 21, Issue 1 (January 1978), pages 4-12

Atlas

Atlas

[O72] The Multics System: An Examination of its Structure Elliott Organick, 1972

Multics

*Fred Brooks*

[B75]

[PP03] Introduction to Computing Systems: From Bits and Gates to C and Beyond

Yale N. Patt and Sanjay J. Patel

McGraw-Hill, 2003

C

[RT74] The UNIX Time-Sharing System Dennis M. Ritchie and Ken Thompson

CACM, Volume 17, Number 7, July 1974, pages 365-375

UNIX

UNIX

[S68] SDS 940 Time-Sharing System Scientific Data Systems Inc.

TECHNICAL MANUAL, SDS 90 11168 August 1968

20

60

SDS

Butler Lampson

[SS+10] Membrane: Operating System Support for Restartable File Systems Swaminathan Sundararaman,  
Sriram Subramanian, Abhishek Rajimwale, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Michael M.  
Swift FAST 10, San Jose, CA, February 2010

Membrane