OS

January 22, 2019

Contents

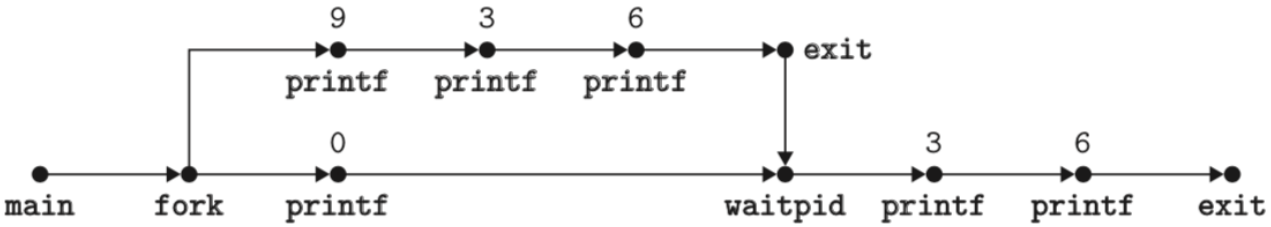
8.3 (page 781)

List all of the possible output sequences for the following program.

```
int main()
{
  if (Fork() == 0) {
    printf("9"); fflush(stdout)
  }
  else {
    printf("0"; fflush(stdout);
    waitpid(-1, NULL, 0);
  }
  printf("3"); fflush(stdout);
  printf("6"); exit(0);
}
```

Answer

A good idea for this kind of task is to make a diagram as created below. This gives a good understanding of how fork works and it is then easier to count all the possible output sequences for the program above.



By looking at the diagram above we can see that all the possible output sequences for the program is

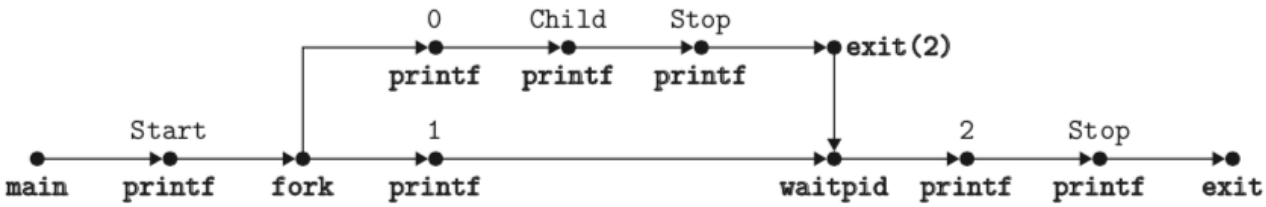
1. 936036
2. 093636
3. 903636

8.4 (page 784)

Consider the following program

```
1   int main()
2   {
3       int status;
4       pid_t pid;
5
6       printf("Start\n");
7       pid = Fork();
8       printf("%d\n", !pid);
9       if (pid == 0) {
10          printf("Child\n");
11      }
12      else if ((waitpid(-1, &status, 0) > 0) &&
                    (WIFEXITED(status) != 0)) {
13          printf("%d\n", WEXITSTATUS(status));
14      }
15      printf("Stop\n");
16      exit(2);
17  }
```

Answer



How many output lines does this program generate?

The output lines is just the number of print statements which in this case is 7

What is one possible ordering of these output lines?

Start, 0, 1, Child, Stop, 2, Stop

8.9 (page 824)

Consider four processes with the following starting and ending times:

Process	Start time	End time
A	6	8
B	3	5
C	4	7
D	2	9

For each pair of processes, indicate whether they run concurrently (Y) or not (N):

To indicate whether they run concurrently it is important to know what that actually means. It just means that you have to check if they at any time, run at the same time.

Answer

Process pair	Concurrent
AB	N
AC	Y
AD	Y
BC	Y
BD	Y
CD	Y

8.12 (page 825)

How many Example output lines does this program print?

```
1   #include "csapp.h"
2
3   void try()
4   {
5       Fork();
6       printf("Example\n");
7       Fork();
8       return;
9   }
10
11  int main()
12  {
13      try(); Fork();
14      printf("Example\n");
15      exit(0);
16  }
```

Answer

The program prints "Example" 10 times.

8.23

9.2 (page 843)

Determine the number of page table entries (PTE_s) that are needed for the following combinations of the virtual adess size(n) and page size(p)

Answer

To compute this we use the formula 2^{n-p} We are given n but to get p we have to take the logarithm K, M, G etc.

n	$p = 2^p$	Computed p	computation	number of PTE_s (page table entries)
12	1K	$p = \log2(2^{10}) = 10$	2^{12-10}	4
16	16K	$p = \log2(2^{10} \cdot 16) = 14$	2^{16-14}	4
24	2M	$p = \log2(2^{20} \cdot 2) = 21$	2^{24-21}	8
36	1G	$p = \log2(2^{30}) = 30$	2^{36-30}	64

9.3 (page 852)

Given a 64-bit virtual address space and a 32-bit physical address, determine the number of bits in the VPN, VPO, PPN, and PPO for the following page sizes P

Answer

VPO and PPO from 0 to $p-1$.
VPN from p to $n-1$.
PPN from p to $m-1$.

Example for 1KB

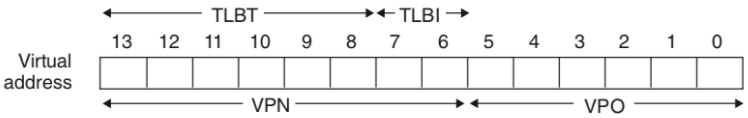
$$n = 64$$
$$m = 32$$
$$p = log_2(1024) = 10$$

VPO and PPO from 0 to $10-1=9$.
VPN from 10 to $64-10=54$.
PPN from 10 to $32-10=22$.

P	VPN bits	VPO bits	PPN bits	PPO bits
1KB	54	10	22	10
2KB	53	11	21	11
4KB	52	12	20	12
16KB	50	14	18	14

Address Translation

- The memory is byte addressable.
- Memory accesses are to 1-byte words (not 4-byte words).
- Virtual addresses are 14 bits wide (n = 14).
- Physical addresses are 12 bits wide (m = 12).
- The page size is 64 bytes (P = 64).
- The TLB is 4-way set associative with 16 total entries.
- The L1 d-cache is physically addressed and direct mapped, with a 4-byte line size and 16 total sets.



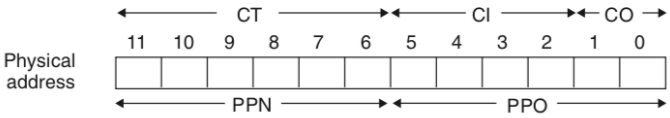
Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

(a) TLB: 4 sets, 16 entries, 4-way set associative

VPN	PPN	Valid
00	28	1
01	—	0
02	33	1
03	02	1
04	—	0
05	16	1
06	—	0
07	—	0

VPN	PPN	Valid
08	13	1
09	17	1
0A	09	1
0B	—	0
0C	—	0
0D	2D	1
0E	11	1
0F	0D	1

(b) Page table: Only the first 16 PTEs are shown



Idx	Tag	Valid	Blk 0	Blk 1	Blk 2	Blk 3
0	19	1	99	11	23	11
1	15	0	—	—	—	—
2	1B	1	00	02	04	08
3	36	0	—	—	—	—
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	—	—	—	—
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	2D	0	—	—	—	—
A	2D	1	93	15	DA	3B
B	0B	0	—	—	—	—
C	12	0	—	—	—	—
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	—	—	—	—

(c) Cache: 16 sets, 4-byte blocks, direct mapped

We start by computing the VPO and VPN.

$$n = 14$$
$$m = 12$$
$$p = log_2(64) = 6$$

VPO

The VPO is computed by using the formula $p - 1$. By doing inserting in the formula we get the following computation.

$$\text{VPO} = 6 - 1 = 5$$

VPN

The VPN is computed by using the formula $n - p$. By doing inserting in the formula we get the following computation.

$$\text{VPN} = 14 - 6 = 8$$

9.4 (page 860)

Answer

A) Virtual address format:

0x03d7 to binary

VPN													
TLBT						TLBI		VPO					
0	0	0	0	1	1	1	1	0	1	0	1	1	1

B) Address translation

Parameter	Value
VPN	0xf
TLBI	0x3
TLBT	0x3
TLB hit?	Yes
Page fault	No
PPN	0xd

C) Physical address format

PPN						PPO					
CT						CI				CO	
0	0	1	1	0	1	0	1	0	1	1	1

D) Physical memory reference

Parameter	Value
CO	0x3
CI	0x5
CT	0xd
Cache hit?	Yes
Cache byte returned	0x1d

9.10 (page 900)

9.11 (page 912)

Answer

A) Virtual address format:

0x027c to binary

VPN													
TLBT						TLBI		VPO					
0	0	0	0	1	0	0	1	1	1	1	1	0	0

B) Address translation

Parameter	Value
VPN	0x9
TLBI	0x1
TLBT	0x2
TLB hit?	No
Page fault	No
PPN	0x17

C) Physical address format

PPN						PPO					
CT						CI				CO	
0	1	0	1	1	1	1	1	1	1	0	0

D) Physical memory reference

Parameter	Value
CO	0x0
CI	0xf
CT	0x17
Cache hit?	No
Cache byte returned	———

9.12 (page 913)

Answer

A) Virtual address format:

0x03a9 to binary

VPN													
TLBT						TLBI		VPO					
0	0	0	0	1	1	1	0	1	0	1	0	0	1

B) Address translation

Parameter	Value
VPN	0xe
TLBI	0x2
TLBT	0x3
TLB hit?	No
Page fault	No
PPN	0x11

C) Physical address format

PPN						PPO					
CT						CI				CO	
0	1	0	0	0	1	1	0	1	0	0	1

D) Physical memory reference

Parameter	Value
CO	0x1
CI	0xa
CT	0x11
Cache hit?	No
Cache byte returned	———

9.13 (page 914)

Answer

A) Virtual address format:

0x0040 to binary

VPN													
TLBT						TLBI		VPO					
0	0	0	0	0	0	0	1	0	0	0	0	0	0

B) Address translation

Parameter	Value
VPN	0x1
TLBI	0x1
TLBT	0x0
TLB hit?	No
Page fault	Yes
PPN	———

9.15 (page 915)

Determine the block sizes and header values that would result from the following sequence of malloc requests. Assumptions: (1) The allocator maintains double-word alignment and uses an implicit free list with the block format from Figure 9.35. (2) Block sizes are rounded up to the nearest multiple of 8 bytes.

Answer

Block size:

We know from the task description that we have to allocate a maintains double-word. And we know that a double-word is the same as 4 bytes. To find the block size we just have to add the maintains double-word (4 bytes) to the number that gets malloced. It is important to know that if you dont get a result that is the exact as 1, 2, 3 etc. bytes we round up to the nearest.

Block header:

To find the block header we just add 1 to the block size and convert this to hex.

Request	Block size (decimal bytes)	Block header (hex)
malloc(4)	$4 + 4 = 8$	0x9
malloc(7)	$4 + 7 = 11$ Round up to nearest byte 16	0x11
malloc(19)	$4 + 19 = 23$ Round up to nearest byte 24	0x19
malloc(22)	$4 + 22 = 26$ Round up to nearest byte 32	0x21

9.16 (page 915)

Determine the minimum block size for each of the following combinations of alignment requirements and block formats. Assumptions: Explicit free list, 4-byte pred and succ pointers in each free block, zero-size payloads are not allowed, and headers and footers are stored in 4-byte words.

Answer

Consider the provided details in the text book:
The headers and footers require 4 bytes. The free block consists of pred and succ pointers of 4 bytes. In the first part, the alignment is of single word; therefore the minimum block size should be the multiple of 4. Allocated block contain header and footer of 4 bytes each along with 1 byte for payload. Therefore its size will be $4 + 4 + 1 = 9$ which is rounded up to 12 bytes.

Free block contains 4 bytes for pred and succ pointers along with 4 bytes for header and footer each. Therefore, its size will be $4 + 4 + 4 + 4 = 16$ which a multiple of 4 is already.

From both the above values 16 is maximum. Hence, the minimum size of the block is 16 bytes.

Alignment	Allocated block	Free block	Minimum block size (bytes)
Single word	Header and footer	Header and footer	$4 + 4 + 4 + 4 = 16$
Single word	Header, but no footer	Header and footer	$4 + 4 + 4 + 4 = 16$
Double word	Header and footer	Header and footer	$4 + 8 + 4 = 16$
Double word	Header, but no footer	Header and footer	$4 + 4 + 4 + 4 = 16$

9.19 (page 916)

You are given three groups of statements relating to memory management and garbage collection below. In each group, only one statement is true. Your task is to indicate which statement is true.

1.

   (a) In a buddy system, up to 50% of the space can be wasted due to internal fragmentation.

   (b) The first-fit memory allocation algorithm is slower than the best-fit algorithm (on average).

   (c) Deallocation using boundary tags is fast only when the list of free blocks is ordered according to increasing memory addresses.

   (d) The buddy system suffers from internal fragmentation, but not from external fragmentation.

2.

   (a) Using the first-fit algorithm on a free list that is ordered according to decreasing block sizes results in low performance for allocations, but avoids external fragmentation.

   (b) For the best-fit method, the list of free blocks should be ordered according to increasing memory addresses.

   (c) The best-fit method chooses the largest free block into which the re- quested segment fits.

   (d) Using the first-fit algorithm on a free list that is ordered according to increasing block sizes is equivalent to using the best-fit algorithm.

3. Mark & Sweep garbage collectors are called conservative if

   (a) They coalesce freed memory only when a memory request cannot be satisfied.

   (b) They treat everything that looks like a pointer as a pointer.

   (c) They perform garbage collection only when they run out of memory.

   (d) They do not free memory blocks forming a cyclic list.