Word Bank for Chapters 11-14

Write one of the words or terms from the following list into the blank appearing to the left of the appropriate definition. Note that there are more words and terms than definitions. (1 pt. each)

<table>
<thead>
<tr>
<th>absolute path</th>
<th>durability</th>
<th>logical record</th>
<th>partition</th>
<th>scrapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>access control list</td>
<td>extent</td>
<td>metadata</td>
<td>physical record</td>
<td>transaction</td>
</tr>
<tr>
<td>capability list</td>
<td>file descriptor</td>
<td>mirroring</td>
<td>relative path</td>
<td>volume</td>
</tr>
<tr>
<td>commit</td>
<td>inode</td>
<td>mount</td>
<td>resident attribute</td>
<td>wear leveling</td>
</tr>
<tr>
<td>device driver</td>
<td>indirect block</td>
<td>open file table</td>
<td>roll back</td>
<td>wear out</td>
</tr>
</tbody>
</table>

1. ______________________ All of the updates occur.

2. ______________________ A value used to refer to an open file.

3. ______________________ State changes survive system crashes.

4. ______________________ The unit of data transfer for an application.

5. ______________________ The unit of data transfer for a physical device.

6. ______________________ The contents are stored directly in the MFT record.

7. ______________________ A path name that is interpreted relative to the root directory.

8. ______________________ A list of (object, access rights) tuples held by a user or application.

9. ______________________ A path name that is interpreted relative to the current working directory.

10. ______________________ A collection of physical storage resources that form a logical storage device.

11. ______________________ A variable-size region of a file that is stored in a contiguous region on a disk.

12. ______________________ The system writes each block of data to two disks and can read any block from either disk.

13. ______________________ A list of (user, access rights) tuples held by an object, which may be stored explicitly or in a compressed format.

14. ______________________ A way to perform a set of updates while providing the properties of atomicity, consistency, isolation, and durability.

15. ______________________ A process, thread, or procedure that translates between the high level abstractions implemented by the operating system and the hardware-specific details of I/O devices.

16. ______________________ Periodically reading the entire contents of a disk, detecting sectors with unrecoverable read errors, reconstructing the lost data, and writing the reconstructed data to spare sectors when the read errors are determined to be permanent.
Kernel mode / User mode. Circle one or both of K and U, as applies. (1 pt. each)
17. K / U The hardware timer can be set in this mode.
18. K / U A store instruction is allowed to execute in this mode.
19. K / U A value from the user stack can be loaded in this mode.
20. K / U All physical memory locations can be accessed in this mode.
21. K / U In a paging system, the Page Table Base Register can be loaded in this mode.

Program / Multithreaded Process / Thread. Circle only one of P, MTP, or T, as applies. (1 pt. each)
22. P / MTP / T The UNIX fork() system call creates an object of this type.
23. P / MTP / T This object has a scheduling state (e.g., running, ready, waiting).
24. P / MTP / T This object has a one-to-one association with a program counter (PC).
25. P / MTP / T This object has a one-to-one association with a user stack pointer (SP).
26. P / MTP / T In a virtual memory paging system, this object has a one-to-one association with a page table.

27. How does the hardware know which interrupt handler to start executing in response to an interrupt? (2 pts.)

28. Consider the diagram below (Figure 2.13 from the textbook). It shows that system call arguments are copied and checked by a kernel stub procedure before the requested operation is performed. A classmate tells you that he thinks that the copying is excessive since the user program is not executing while the system call is being executed and that the arguments on the user stack can instead be checked in place. Explain what you would tell your classmate to change his mind. How could arguments left on the user stack be changed in the time between the kernel stub checks them and the kernel routine uses them? (3 pts.)

```
User Program
main () {
    file_open(arg1, arg2);
}

Kernel
file_open(arg1, arg2) {
    // do operation
}

User Stub
file_open(arg1, arg2) {
    push #SYSCALL_OPEN
    trap
    return
}

(2) Hardware Trap

Kernel Stub
file_openHandler() {
    // copy arguments
    // from user memory
    // check arguments
    file_open(arg1, arg2);
    // copy return value
    // into user memory
    return;
}

(3) (4)
(5)
```
29. Why does an operating system need more than interrupt disable and enable to implement a lock? (3 pts.)

30. Why does an operating system need to provide condition variables or equivalent synchronization primitive? (3 pts.)

31. Identify and briefly explain a benefit of RR (round robin) scheduling over FIFO. (2 pts.)

32. Is RR a preemptive policy or a non-preemptive policy? (1 pt.)

33. Does RR use a FIFO queuing discipline or a sorted queuing discipline? (1 pt.)

34. When would RR (round robin) produce the same exact results as FIFO? (1 pt.)

35. Some systems allow a signal operation to include a temporary scheduling priority boost. Explain the purpose of this boost. (2 pts.)
36. Let the three threads shown below run with static scheduling priority on a uniprocessor. The lock being used is a queueing lock. Explain, perhaps using a timing diagram, the sequence of events and actions that results in priority inversion. You can use the statement identifiers in your explanation, but a list of the statement identifiers by themselves is not sufficient. (4 pts.)

<table>
<thead>
<tr>
<th>low priority thread A</th>
<th>medium-priority thread B</th>
<th>high-priority thread C</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>A1: lock.acquire();</td>
<td>B1: work</td>
<td>C1: lock.acquire();</td>
</tr>
<tr>
<td>A2: critical section</td>
<td>...</td>
<td>C2: critical section</td>
</tr>
<tr>
<td>A3: lock.release();</td>
<td>...</td>
<td>C3: lock.release();</td>
</tr>
</tbody>
</table>

37. What is the difference between spatial and temporal locality? (3 pts.)

Consider a simple system with segmentation in which virtual addresses are four decimal digits and physical addresses are also four decimal digits. A virtual address is composed of a two-digit segment id and a two-digit offset. Give the physical addresses for the following virtual addresses or any other actions taken and why. (1 pt. each)

<table>
<thead>
<tr>
<th>Segment</th>
<th>Base Address</th>
<th>Length</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>3000</td>
<td>50</td>
<td>Read-only</td>
</tr>
<tr>
<td>01</td>
<td>6000</td>
<td>80</td>
<td>Read and Write</td>
</tr>
<tr>
<td>XX (other values are invalid)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

38. On a read operation, virtual address 0060.

39. On a read operation, virtual address 0140.

40. On a write operation, virtual address 0020.
41. Assume a program sequentially traverses a large array of double words using a stride of 8 bytes in virtual addresses on a paging system, with no other data accesses interspersed in this phase of program execution. What kind of access pattern is presented to the entries in the data TLB during this phase? Does the access pattern exhibit spatial, temporal locality, or both? Explain your answer. (2 pts.)

42. Explain copy-on-write and how a paging system would implement it. (4 pts.)

One address translation scheme for the DEC Alpha used a 43-bit virtual address with three levels of page tables. The virtual address was divided into the following fields (the numbers are the field lengths in bits):

<table>
<thead>
<tr>
<th>level 1</th>
<th>level 2</th>
<th>level 3</th>
<th>page offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>13</td>
</tr>
</tbody>
</table>

Answer questions 43-44 using powers of 2 for the address format above. Use bytes as the addressable units.

43. How big is a page in bytes? (1.5 pts.)

44. If a page table entry is 8 bytes in size, how big is a page table in bytes? (1.5 pts.)
Directory Entry / Inode / Per-Open Data Structure. Circle **only one** of DE, I, or PODS. Assume an FFS file. (2 pts. each)

45. DE / I / PODS  Contains the file metadata.
46. DE / I / PODS  Contains pointers to indirect blocks.
47. DE / I / PODS  Contains the pointer to the current byte or record in a file.
48. DE / I / PODS  The data structure that supports a connection-oriented interface for file accesses.

49. Label the following steps that occur in opening an existing file in their proper sequential order, 1-3. (1 pt. each)

   ____ Initialize the file position pointer in the per-open data structure to the first byte (or record) of the file.
   ____ Find the directory entry for the named file. Check the access permissions and return an error code if the requested access is not allowed.
   ____ Create a process-local per-open data structure and record the access permission under which the file was opened and the location of the file’s inode (or other location information).

50. When a computer system starts, non-volatile storage devices are accessed, such as a ROM and then a hard disk. What type of information is obtained from the non-volatile devices? (2 pts.)

51. What are two possible problems with this code pattern? (4 pts.)

   ```
   if( !exists( file ) ){
       create( file );
   }
   open( file );
   ```

52. How does logical record blocking provide better I/O system performance? (2 pts.)
53. The time required for a sequence of five reads to random sectors on a disk can take almost five times longer than the time required to read five contiguous sectors. What are the factors involved in disk access that lead to such a disparity? (3 pts.)

54. In the two columns below, identify the set of steps needed to access byte 5000 in a file organized using a FAT and a file organized as FFS. State the steps in such a way that you make it clear that you know the differences between the two file organizations based on the textbook examples. (There is no need to include vnode processing for FFS.) Assume the file block size is 4 KiB. (4 pts.)

<table>
<thead>
<tr>
<th>FAT</th>
<th>FFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) set the current byte offset to 5000</td>
<td>1) set the current byte offset to 5000</td>
</tr>
<tr>
<td>2) obtain the index of first file block of the file from the directory entry</td>
<td>2) obtain the file block number of the inode of the file from the directory entry</td>
</tr>
</tbody>
</table>

For questions 55-57, consider the reference count field in an inode.

55. What are the “references” that are being counted? (1 pt.)

56. When is the reference count decremented? (1 pt.)

57. What does the file system do when the reference count reaches 0? (1 pt.)
58. Directories in modern file systems are often structured as balanced trees. For example, the Linux XFS file system uses a B+ tree. Why was FFS organized as an asymmetric tree of block pointers rather than as a balanced tree? (2 pts.)

59. Explain how a CoW file system optimizes write performance. (2 pts.)

60. Label four steps below that occur in a redo logging transaction in proper sequential order, 1-4. Note that there are more steps listed below than are appropriate. (4 pts.)

____ Make a copy of the log.
____ Recover and roll back the updates.
____ Append a commit record to the log.
____ Write the updates to target locations.
____ Garbage collect the records in the log.
____ Append all planned updates to the log.
____ Tag the target locations with a commit bit.
____ Read the values back from the log and check them.
____ Write the old values of the target locations to the log.