Exam 27 June 2019:

1. Write a UNIX command that displays all the lines in the a.txt file that contain at least one binary number that is a multiple of 4 with 5 or more digits (ex: 010100)

```
grep -E '\b[01]{3,}00\b' a.txt
```

2. Write a UNIX command that reverses all pairs of odd numbers followed by vowels (ex: a2328i97u3 → a2238i9u73)

```
sed -E 's/([13579])([aeiou])/\2\1/g'
```

3. Write a UNIX commamnd that displays all the unique soccer scores (ex: 4-0) that appear in the a.txt file. The number of goals can have a maximum of two digits.

```
grep -o -E '[0-9]{1,2}-[0-9]{1,2}' a.txt | sort | uniq
```

4. Display the number of processes of each active user in the system.

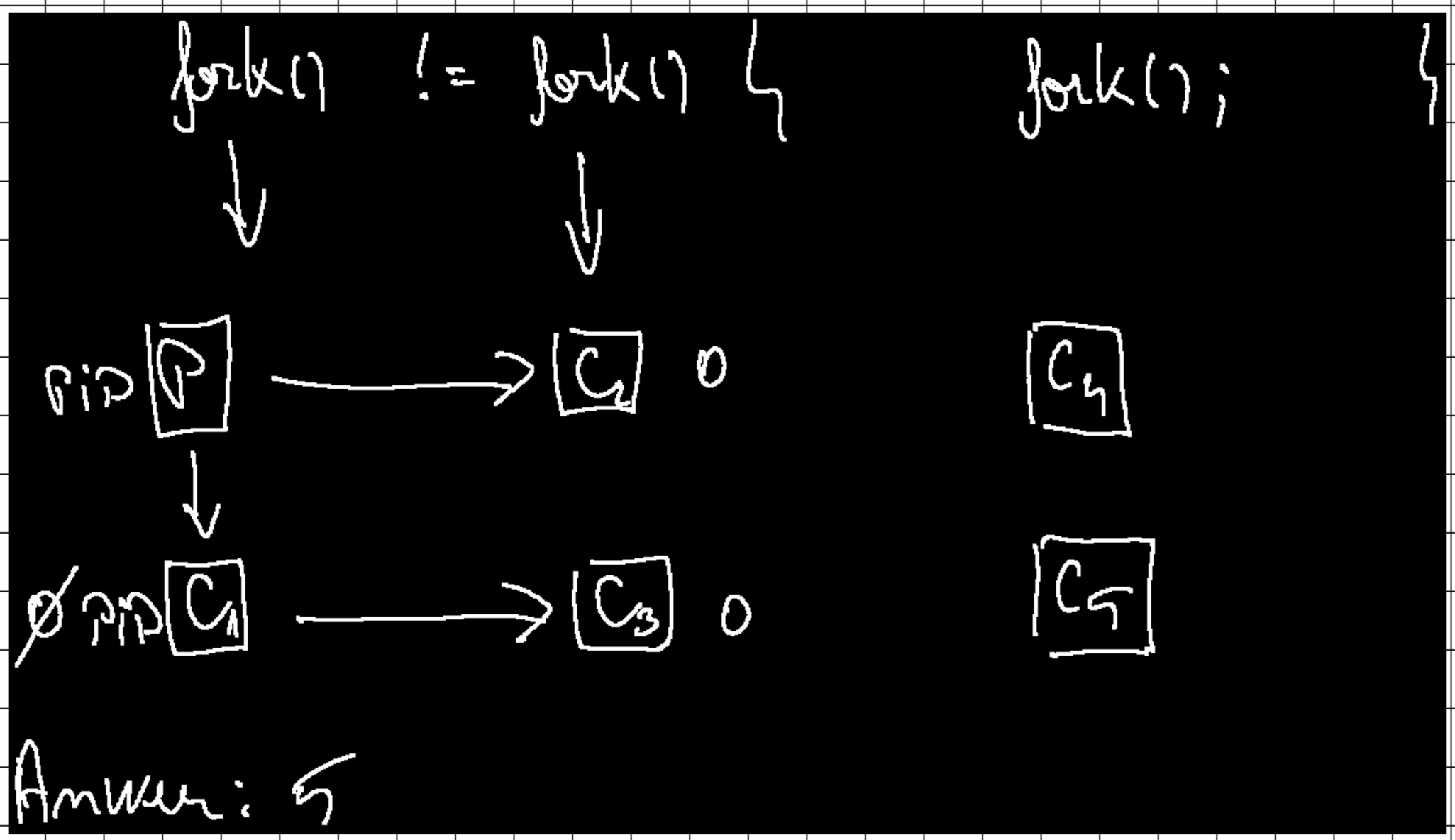
```
ps -eo user= | sort | uniq -c
```

5. Write a UNIX shell script that calculates the average number of .txt files per directory in the current directory and all it's subdirectories.

```
#!/bin/bash

total_txt_files=$(find . -type f -name "*.txt" | wc -l)

total_dirs=$(find . -type d | wc -l)

if [ $total_dirs -eq 0 ]; then
    average=0
else
    average=$(( total_txt_files / total_dirs ))
fi

echo "$average"
```

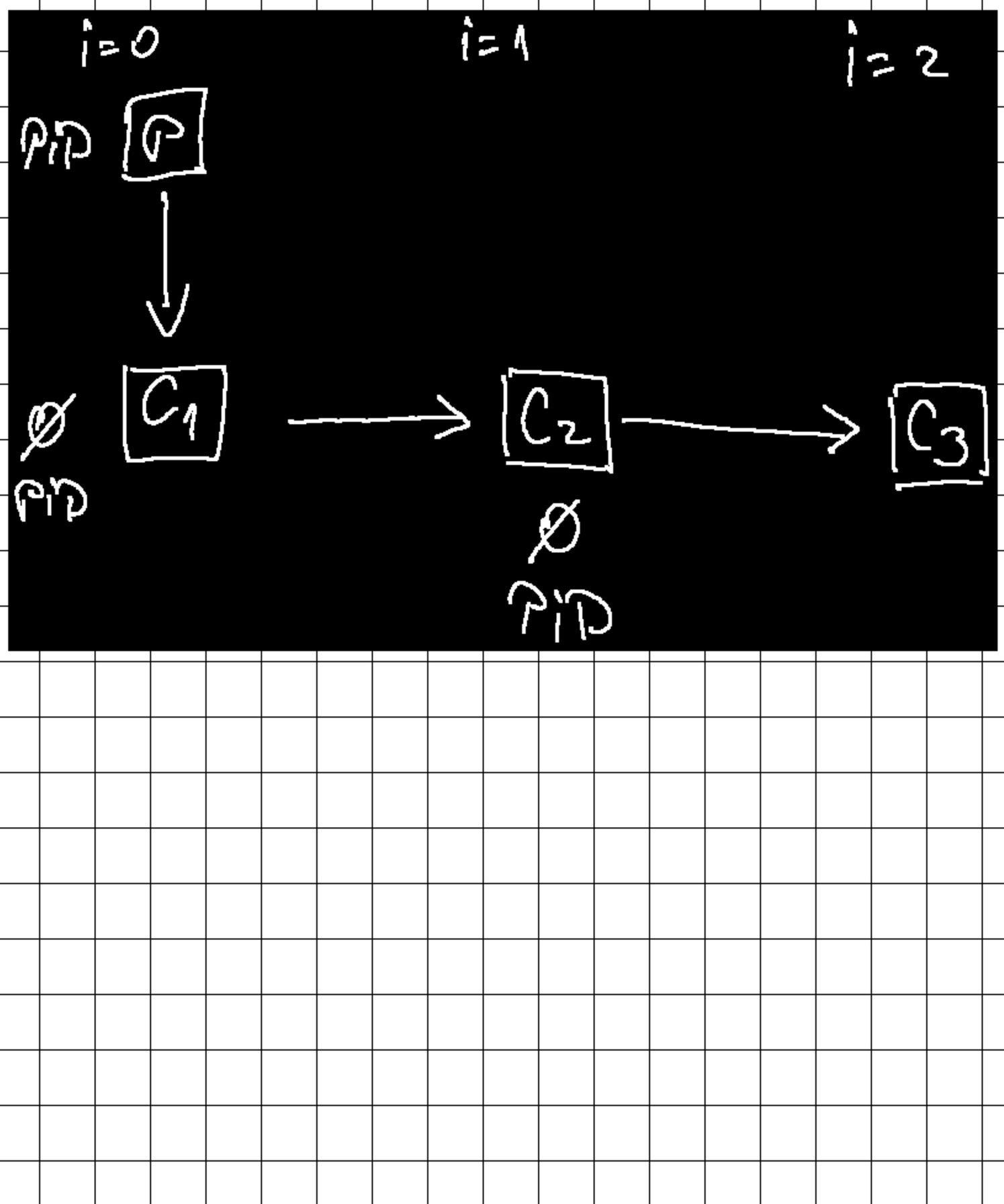
6. How many processes will the code below create, excluding the initial parent process?

```
if ( fork() != fork() ) {
        fork();
}
```



7. Draw the process hierarchy generated by the code below

```
int p=0;
for(int i=0; i<3; i++)
{
    if(p==0)
    p=fork();
else
    wait(0);
}
```



8. What does the code below print in the console

```
char *S[3] = {"A", "B", "C"};
for (int i=0; i<3; i++){
    if (fork()==0){
        execl("/bin/echo", "/bin/echo", S[i], NULL);
    }
}
```

Answer: A B C

9. What does the "write" system call do when there is space in the PIPE, but not enough for what it asks to write?

Answer:

The "write" system call will block, the call will wait until enough space becomes available in the PIPE to write the entire data, and "write" will return the number of bytes written

10. What does the code below print if no other process opens the "abc" FIFO? Justify the answer.

```
int w, n, K=10;
n= open("abc", O_WRONLY);
n = write(n, &K, sizeof(int));
printf("%d\n", n);
```

Answer: – The "open" call will block until another process opens the FIFO in O_RDONLY
– no other process opens the FIFO "abc" (stated in the pb.
– the "write" and "printf" call will never be executed
– nothing will be printed

11. What happens to zombie processes whose parent has terminated?

Answer:
– When the parent of a zombie process terminates, it becomes an orphan
– it gets adopted by the "init" process that periodically calls wait() to clean up orphaned zombies.

2. Consider that function f is executed simultaneously by 10 threads. Add the necessary lines of code to ensure that n will have the value 10 after the threads finish executing

```
int n=0
pthread_mutex_t mutex;
void *f (void *p)
{
    pthread_mutex_lock (& mutex);
    n++;
    pthread_mutex_unlock (& mutex);
    return NULL;
}
```

13. Plan the execution of the following jobs (given as Name / Duration / Deadline) so that the amount of job delays is minimal:

A/22/27, B/2/15, C/4/15

Job C/4/15

Start time: 0

end time: 4 no delay

Job B/2/15

Start time: 4

end time: 6 no delay

Job A/22/27

start time: 6

end time: 28 ⇒ 28-27 = 1 delay

Total delay is 1 which is minimal

4. Give an advantage and a disadvantage of set-associative caches compared to direct ones.

Advantage: reduced conflict misses, allows more flexibility in block placement within a set, reducing the likelihood of conflict misses compared to direct ones.
Disadvantage: Increased complexity and cost.