Labs

Neo Pi

Neo.Pi1@Marist.edu

September 9, 2024

1 Lab One

1.1 What are the advantages and disadvantages of using the same system call interface for manipulating both files and devices?

The advantage of using the same system call interface for both files and devices is so that each device can be accessed as thought it was a file within the system. Since kernel already deals with devices through a file interface, adding code to support a different kind of file interface is possible. The disadvantage of using the same interface is that the functionality of some devices might not be able to be accessed. Loss of functionality or performance could be a result.

1.2 Would it be possible for the user to develop a new command interpreter using the system call interface provide by the operating system? How?

Yes, a user can develop a new command interpreter by using the system call interface provided by the operating system. You can develop it by specifically using system calls to manage processes and file-related system calls for i/o operations.

2 Lab Two

2.1 How is your console like the ancient TTY subsystem in Unix as described in https://www.linusaskesson/programming/tty/?

Originally, the TTY system was designed for text-based input and output. Similarly, the current console we are working on operates the same way. It takes character-based commands, and handles commands by processing input as text characters. The input is buffered line-by-line, just as it is in our own OS. Anohter important factor, is that the TTV subsystem was the primary interface for the system, and similarly our OS provides a similar interactive interface. Lastly, the interaction is all in real time, when issuing commands and receiving feedback.

3 Problem Two

4 Problem Three

References

5 Appendix

5.1 Some JavaScript source code listings

```
var A = [5,0,5,6,6,8,45,50];

function solve(A) {
    // Base case to stop the recursion.
    if (A.length == 1) {
        if (A[0] % 5 == 0) {
            var retVal = 1;
        } else {
            var retVal = 0;
        }
        return retVal;
    } else {
        // Divide.
        var divPoint = Math.floor( A.length / 2);
        var left  = A.slice(0, divPoint);
        var right = A.slice(divPoint, A.length);

        // Conquer.
        var left5s   = solve(left, level+1);
        var center5s = straddle(left, right);
        var right5s  = solve(right, level+1);

        // Combine.
        return Math.max(left5s, Math.max(center5s, right5s));
    }
}

function straddle(left, right) {
    var retVal = 0;
    if ((left[left.length-1] % 5 == 0) && (right[0] % 5 == 0)) {
        // Count back the 5's on the left going from right to left.
        var leftCount = 0;
        var index = left.length-1;
        while ( (index >= 0) && (left[index] % 5 == 0) ) {
            index--;
            leftCount++;
        }
        // Count forward the 5's on the right going from left to right.
        var rightCount = 0;
        while ( (rightCount < right.length) && (right[rightCount] % 5 == 0) ) {
            rightCount++;
        }
        // Return the sum of the straddling 5s on the left and right.
        retVal = leftCount + rightCount;
    }
    return retVal;
}
```