

**IT314-Software Engineering LAB-7**

Name: Stavan Ravisaheb

Student ID: 202201436

Lab Group: 5

**|| . Code Debugging and Program Inspection of the JAVA files.**

**1 . Armstrong**

# A . Program Inspection

1. There is one error in the program, related to the computation of the remainder, and it has been identified and corrected.
2. The most effective category of program inspection for this code is Category C: Computation Errors, as the error pertains to the computation of the remainder, a type of computation error.
3. Program inspection does not identify debugging-related errors. It does not detect issues such as breakpoints or runtime errors like logic errors.
4. The program inspection technique is valuable for identifying and rectifying issues related to code structure and computation errors.

# B. Debugging

1. There is one error in the program related to the computation of the remainder, as previously identified.
2. To fix this error, one should set a breakpoint at the point where the remainder is computed to ensure it’s calculated correctly. Step through the code to observe the values of variables and expressions during execution.
3. The corrected executable code is as follows: // Armstrong Number class Armstrong { public static void main(String args[]) { int num = Integer.parseInt(args[0]); int n = num; // used to check at the last time int check = 0, remainder; while (num > 0) { remainder = num % 10; check = check + (int) Math.pow(remainder, 3); num = num / 10;

}

if (check == n)

System.out.println(n + " is an Armstrong Number"); else

System.out.println(n + " is not an Armstrong Number");

}

}

# GCD and LCM

## A. Program Inspection

1. There are two errors in the program:
2. Error 1: In the gcd function, the while loop condition should be while(a % b != 0) instead of while(a % b == 0) to calculate the GCD correctly.
3. Error 2: In the lcm function, there is a logic error. The logic used to calculate LCM is incorrect and will result in an infinite loop.
4. For this code, the most effective category of program inspection is Category C: Computation Errors, as it contains computation errors in both the gcd and lcm functions.
5. Program inspection is not able to identify runtime issues or logical errors. It can’t identify errors like infinite loops.
6. The program inspection technique is worth applying to identify and fix computation-related issues.

## B. Debugging

1. There are two errors in the program as mentioned above.
2. To fix these errors:
3. For Error 1 in the gcd function, you need one breakpoint at the beginning of the while loop to verify the correct execution of the loop.
4. For Error 2 in the lcm function, you would need to review the logic for calculating LCM, as it’s a logical error.
5. The corrected executable code is as follows:

import java.util.Scanner; public class GCD\_LCM { static int gcd(int x, int y) { int a, b; a = (x > y) ? x : y; // a is greater number b = (x < y) ? x : y; // b is smaller number

while (b != 0) { // Fixed the while loop condition int temp = b; b = a % b; a = temp;

}

return a;

}

static int lcm(int x, int y) { return (x \* y) / gcd(x, y); // Calculate LCM using GCD

}

public static void main(String args[]) {

Scanner input = new Scanner(System.in); System.out.println("Enter the two numbers: ");

int x = input.nextInt(); int y = input.nextInt();

System.out.println("The GCD of two numbers is: " + gcd(x, y)); System.out.println("The LCM of two numbers is: " + lcm(x, y)); input.close();

}

}

# Knapsack

## A. Program Inspection

1. There is one error in the program. It is in the following line: int option1 = opt[n++][w]; The variable n is incremented, which is not intended. It should be: int option1 = opt[n][w];
2. The category of program inspection that would be most effective for this code is Category C: Computation Errors, as the identified error is related to computation within loops.
3. Program inspection is not able to identify runtime errors or logical errors that might arise during program execution.
4. The program inspection technique is worth applying to identify and fix computation-related issues.

## B. Debugging

1. There is one error in the program, as identified above.
2. To fix this error, you would need one breakpoint at the line: int option1 = opt[n][w]; to ensure n and w are correctly used without unintended increments.
3. The corrected executable code is as follows: public class Knapsack { public static void main(String[] args) { int N = Integer.parseInt(args[0]); // number of items int W = Integer.parseInt(args[1]); // maximum weight of knapsack

int[] profit = new int[N + 1]; int[] weight = new int[N + 1]; // Generate random instance, items 1..N for (int n = 1; n <= N; n++) { profit[n] = (int) (Math.random() \* 1000); weight[n] = (int) (Math.random() \* W);

}

int[][] opt = new int[N + 1][W + 1]; boolean[][] sol = new boolean[N + 1][W + 1];

for (int n = 1; n <= N; n++) { for (int w = 1; w <= W; w++) { int option1 = opt[n - 1][w]; // Fixed the increment here int option2 = Integer.MIN\_VALUE;

if (weight[n] <= w) option2 = profit[n] + opt[n - 1][w - weight[n]];

opt[n][w] = Math.max(option1, option2); sol[n][w] = (option2 > option1);

}

}

System.out.println("Item" + "\t" + "Profit" + "\t" + "Weight" + "\t" + "Take");

for (int n = 1; n <= N; n++) {

System.out.println(n + "\t" + profit[n] + "\t" + weight[n] + "\t" + take[n]);

}

}

}

# Magic Number

## A. Program Inspection

1. There are two errors in the program:
2. Error 1: In the inner while loop, the condition should be while (sum > 0) instead of while (sum == 0).
3. Error 2: Inside the inner while loop, there are missing semicolons in the lines: s=s\*(sum/10); sum=sum%10; They should be corrected as: s = s \* (sum / 10); sum = sum % 10;
4. The category of program inspection that would be most effective for this code is Category C: Computation Errors, as it contains computation errors in the while loop.
5. Program inspection is not able to identify runtime issues or logical errors that might arise during program execution.
6. The program inspection technique is worth applying to identify and fix computation-related issues.

## B. Debugging

1. There are two errors in the program, as identified above.
2. To fix these errors, you would need one breakpoint at the beginning of the inner while loop to verify the execution of the loop. You can also use breakpoints to check the values of num and s during execution.
3. The corrected executable code is as follows:

import java.util.\*;

public class MagicNumberCheck { public static void main(String args[]) {

Scanner ob = new Scanner(System.in);

System.out.println("Enter the number to be checked.");

int n = ob.nextInt();

int sum = 0, num = n;

while (num > 9) { sum = num;

int s = 0;

while (sum > 0) { // Fixed the condition here s = s \* (sum / 10); sum = sum % 10; // Fixed the missing semicolon

}

num = s;

}

if (num == 1) {

System.out.println(n + " is a Magic Number.");

} else {

System.out.println(n + " is not a Magic Number.");

}

}

}

# Merge Sort

## A. Program Inspection

1. There are several errors in the program:
2. Error 1: In the mergeSort method, the lines int[] left = leftHalf(array+1); and int[] right = rightHalf(array-1); should be corrected. It seems like an attempt to split the array, but it’s not done correctly.
3. Error 2: The leftHalf and rightHalf methods are incorrect. They should return the correct halves of the array.
4. Error 3: The merge method should have left and right arrays as inputs, not left++ and right--.
5. The category of program inspection that would be most effective for this code is Category C: Computation Errors, as there are computation-related issues in the code.
6. Program inspection cannot identify runtime issues or logical errors that might arise during program execution.
7. The program inspection technique is worth applying to identify and fix computation-related issues.

## B. Debugging

1. There are multiple errors in the program, as identified above.
2. To fix these errors, you would need to set breakpoints to examine the values of left, right and array during execution. You can also use breakpoints to check the values of i1 and i2 inside the merge method.
3. The corrected executable code is as follows:

import java.util.\*; public class MergeSort { public static void main(String[] args) { int[] list = {14, 32, 67, 76, 23, 41, 58, 85};

System.out.println("before: " + Arrays.toString(list));

mergeSort(list);

System.out.println("after: " + Arrays.toString(list));

}

public static void mergeSort(int[] array) { if (array.length > 1) { int[] left = leftHalf(array); int[] right = rightHalf(array);

mergeSort(left); mergeSort(right); merge(array, left, right);

}

}

public static int[] leftHalf(int[] array) { int size1 = array.length / 2; int[] left = new int[size1];

for (int i = 0; i < size1; i++) { left[i] = array[i];

}

return left;

}

public static int[] rightHalf(int[] array) { int size1 = array.length / 2; int size2 = array.length - size1; int[] right = new int[size2];

for (int i = 0; i < size2; i++) { right[i] = array[i + size1];

}

return right;

}

public static void merge(int[] result, int[] left, int[] right) { int i1 = 0; int i2 = 0;

for (int i = 0; i < result.length; i++) { if (i2 >= right.length || (i1 < left.length && left[i1] <= right[i2])) { result[i] = left[i1]; i1++;

} else { result[i] = right[i2]; i2++;

}

}

}

}

# Multiply Matrices

## A. Program Inspection

1. There are several errors in the program:
2. Error 1: In the nested loops for matrix multiplication, the loop indices should start from 0, not -1.
3. Error 2: The error message when the matrix dimensions are incompatible should print ”Matrices with entered orders can’t be multiplied with each other,” not ”Matrices with entered orders can’t be multiplied with each other.”
4. The category of program inspection that would be most effective for this code is Category C: Computation Errors, as there are computation-related issues in the code.
5. Program inspection cannot identify runtime issues or logical errors that might arise during program execution.
6. The program inspection technique is worth applying to identify and fix computation-related issues.

## B. Debugging

1. There are multiple errors in the program, as identified above.
2. To fix these errors, you would need to set breakpoints to examine the values of c, d, k, and sum during execution. You should pay particular attention to the nested loops where the matrix multiplication occurs.
3. The corrected executable code is as follows:

import java.util.Scanner; class MatrixMultiplication { public static void main(String args[]) { int m, n, p, q, sum = 0, c, d, k;

Scanner in = new Scanner(System.in);

System.out.println("Enter the number of rows and columns of the first

matrix");

m = in.nextInt(); n = in.nextInt(); int first[][] = new int[m][n];

System.out.println("Enter the elements of the first matrix");

for (c = 0; c < m; c++) for (d = 0; d < n; d++)

first[c][d] = in.nextInt();

System.out.println("Enter the number of rows and columns of the second matrix"); p = in.nextInt(); q = in.nextInt();

if (n != p)

System.out.println("Matrices with entered orders can't be multiplied with each other."); else { int second[][] = new int[p][q]; int multiply[][] = new int[m][q];

System.out.println("Enter the elements of the second matrix");

for (c = 0; c < p; c++) for (d = 0; d < q; d++)

second[c][d] = in.nextInt();

for (c = 0; c < m; c++) { for (d = 0; d < q; d++) { for (k = 0; k < p; k++) { sum = sum + first[c][k] \* second[k][d];

}

multiply[c][d] = sum; sum = 0;

}

}

System.out.println("Product of entered matrices:-");

for (c = 0; c < m; c++) { for (d = 0; d < q; d++)

System.out.print(multiply[c][d] + "\t");

System.out.print("\n");

}

}

}

}

# Quadratic Probing

## A. Program Inspection

1. There are multiple errors in the program:
2. Error 1: The insert method has a typo in the line i + = (i + h / h–)
3. Error 2: In the remove method, there is a logic error in the loop to rehash keys. It should be i = (i + h \* h++)
4. Error 3: In the get method, there is a logic error in the loop to find the key. It should be i = (i + h \* h++)
5. The category of program inspection that would be most effective for this code is Category A: Syntax Errors and Category B: Semantic Errors, as there are both syntax errors and semantic issues in the code.
6. The program inspection technique is worth applying to identify and fix these errors, but it may not identify logical errors that affect the program’s behaviour.

## B. Debugging

1. There are three errors in the program, as identified above.
2. To fix these errors, you would need to set breakpoints and step through the code while examining variables like i, h, tmp1, and tmp2. You should pay attention to the logic of the insert, remove and get methods.
3. The corrected executable code is as follows:

import java.util.Scanner; class QuadraticProbingHashTable { private int currentSize, maxSize;

private String[] keys; private String[] vals;

public QuadraticProbingHashTable(int capacity) { currentSize = 0; maxSize = capacity; keys = new String[maxSize]; vals = new String[maxSize];

}

public void makeEmpty() { currentSize = 0; keys = new String[maxSize]; vals = a String[maxSize];

}

public int getSize() { return currentSize;

}

public boolean isFull() { return currentSize == maxSize;

}

public boolean isEmpty() { return getSize() == 0;

}

public boolean contains(String key) { return get(key) != null;

}

private int hash(String key) { return key.hashCode() % maxSize; public void insert(String key, String val) { int tmp = hash(key); int i = tmp, h = 1;

do {

if (keys[i] == null) { keys[i] = key; vals[i] = val; currentSize++;

return;

}

if (keys[i].equals(key)) { vals[i] = val;

return;

}

i += (h \* h++) % maxSize;

} while (i != tmp);

}

public String get(String key) { int i = hash(key), h = 1;

while (keys[i] != null) { if (keys[i].equals(key)) return vals[i];

i = (i + h \* h++) % maxSize;

return null;

}

public void remove(String key) { if (!contains(key)) return; int i = hash(key), h = 1;

while (!key.equals(keys[i]))

i = (i + h \* h++) % maxSize;

keys[i] = vals[i] = null; for (i = (i + h \* h++) % maxSize; keys[i] != null; i = (i + h \* h++) % maxSize)

{

String tmp1 = keys[i], tmp2 = vals[i]; keys[i] = vals[i] = null; currentSize--; insert(tmp1, tmp2);

}

currentSize--;

}

public void printHashTable() {

System.out.println("\nHash Table: ");

for (int i = 0; i < maxSize; i++) if (keys[i] != null)

System.out.println(keys[i] + " " + vals[i]);

System.out.println();

}

public class QuadraticProbingHashTableTest { public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.println("Hash Table Test\n\n");

System.println("Enter size");

QuadraticProbingHashTable qpht = new QuadraticProbingHashTable(scan.nextInt()); char ch;

do {

System.out.println("\nHash Table Operations\n");

System.out.println("1. insert");

System.out.println("2. remove");

System.out.println("3. get");

System.out.println("4. clear"); System.out.println("5. size"); int choice = scan.nextInt();

switch (choice) { case 1:

System.out.println("Enter key and value"); qpht.insert(scan.next(), scan.next());

Break; case 2:

System.out.println("Enter key"); qpht.remove(scan.next());

Break;

case 3:

System.out.println("Enter key");

System.out.println("Value = " + qpht.get(scan.next()));

Break; case 4:

qpht.makeEmpty();

System.out.println("Hash Table Cleared\n");

Break; case 5:

System.out.println("Size = " + qpht.getSize());

Break; default:

System.out.println("Wrong Entry\n"); break;

}

qpht.printHashTable();

System.out.println("\nDo you want to continue (Type y or n) \n"); ch = scan.next().charAt(0);

} while (ch == 'Y' || ch == 'y');

}

}

# Sorting Array

A. Program Inspection

1. Errors identified:
2. Error 1: The class name ”Ascending Order” contains an extra space and an underscore. The class name should be corrected to ”AscendingOrder.”
3. Error 2: The first nested for loop has an incorrect loop condition for (int i = 0; i ¿= n; i++);, which should be modified to for (int i = 0; i ¡ n; i++).
4. Error 3: There is an extra semicolon (;) after the first nested for loop, which should be removed.
5. The most effective category of program inspection would be Category A: Syntax Errors and Category B: Semantic Errors, as there are both syntax errors and semantic issues in the code.
6. Program inspection alone can identify and fix syntax errors and some semantic issues. However, it may not detect logic errors that affect the program’s behavior.
7. The program inspection technique is worth applying to fix the syntax and semantic errors, but debugging is required to address logic errors.

## B. Debugging

1. There are two errors in the program as identified above.
2. To fix these errors, you need to set breakpoints and step through the code.

You should focus on the class name, the loop conditions, and the unnecessary semicolon.

1. The corrected executable code is as follows:

import java.util.Scanner; public class AscendingOrder { public static void main(String[] args) { int n, temp;

Scanner s = new Scanner(System.in);

System.out.print("Enter the number of elements you want in the array: "); n = s.nextInt(); int a[] = new int[n];

System.out.println("Enter all the elements:");

for (int i = 0; i < n; i++) { a[i] = s.nextInt();

}

for (int i = 0; i < n; i++) { for (int j = i + 1; j < n; j++) {

if (a[i] > a[j]) { temp = a[i]; a[i] = a[j];

a[j] = temp;

}

}

}

System.out.print("Ascending Order: "); for (int i = 0; i < n - 1; i++) {

System.out.print(a[i] + ", ");

}

System.out.print(a[n - 1]);

}

}

# Stack Implementation

## A. Program Inspection

1. Errors identified:
2. Error 1: The push method has a decrement operation on the top variable (top–) instead of an increment operation. It should be corrected to top++ to push values correctly.
3. Error 2: The display method has an incorrect loop condition in for(int i=0; i ¿ top; i++). The loop condition should be for (int i = 0; i ¡= top; i++) to correctly display the elements.
4. Error 3: The pop method is missing in the StackMethods class. It should be added to provide a complete stack implementation.
5. The most effective category of program inspection would be Category A:

Syntax Errors, as there are syntax errors in the code. In addition, Category B:

Semantic Errors can help identify logic and functionality issues.

1. The program inspection technique is worth applying to identify and fix syntax errors, but additional inspection is needed to ensure the logic and functionality are correct.

## B. Debugging

1. There are three errors in the program, as identified above.
2. To fix these errors, you would need to set breakpoints and step through the code, focusing on the push, pop, and display methods. Correct the push and display methods and add the missing pop method to provide a complete stack implementation.
3. The corrected executable code is as follows:

public class StackMethods { private int top; int size; int[] stack;

public StackMethods(int arraySize) { size = arraySize; stack = new int[size]; top = -1;

}

public void push(int value) { if (top == size - 1) {

System.out.println("Stack is full, can't push a value");

} else { top++; stack[top] = value;

}

}

public void pop() { if (!isEmpty()) { top--;

} else {

System.out.println("Can't pop...stack is empty");

}

}

public boolean isEmpty() { return top == -1;

}

public void display() { for (int i = 0; i <= top; i++) {

System.out.print(stack[i] + " ");

}

System.out.println();

}

}

# Tower of Hanoi

## A. Program Inspection

1. Errors identified:
2. Error 1: In the line doTowers(topN ++, inter–, from+1, to+1), there are errors in the increment and decrement operators. It should be corrected to doTowers(topN - 1, inter, from, to).
3. The most effective category of program inspection would be Category B: Semantic Errors because the errors in the code are related to logic and function.
4. The program inspection technique is worth applying to identify and fix semantic errors in the code.

## B. Debugging

1. There is one error in the program, as identified above.
2. To fix this error, you need to replace the line: doTowers(topN ++, inter--, from+1, to+1);
3. with the correct version:

doTowers(topN - 1, inter, from, to);

1. The corrected executable code is as follows:

public class MainClass { public static void main(String[] args) { int nDisks = 3;

doTowers(nDisks, 'A', 'B', 'C');

}

public static void doTowers(int topN, char from, char inter, char to) { if (topN == 1) {

System.out.println("Disk 1 from " + from + " to " + to);

} else { doTowers(topN - 1, from, to, inter);

System.out.println("Disk " + topN + " from " + from + " to " + to); doTowers(topN - 1, inter, from, to);

}

}

}

**| . Program**

**Inspection/Debugging for Long-code from GitHub**

**We are given the following checklist and we have to find all the possible errors accordingly,**

1. Data referencing Errors
2. Data declaration Errors
3. Computation Errors
4. Comparison Errors
5. Control Flow errors
6. Interface errors
7. Input/Output Errors
8. Other Checks

**First code is a Node JS file from**

https://github.com/Medium/medium-sdk-nodejs/blob/master/test/mediumClient\_test.js

The first half of the code is provided here ::

|  |
| --- |
| var medium = require("../") |
| var nock = require("nock") |
| var qs = require('querystring') |
| var should = require("should") |
| var url = require('url') |
|  |
| describe('MediumClient - constructor', function () { |
|  |
| it('should throw a MediumError when options are undefined', function (done) { |
| (function () { new medium.MediumClient() }).should.throw(medium.MediumError) |
| done() |
| }) |
|  |
| it('should throw a MediumError when options are empty', function (done) { |
| (function () { new medium.MediumClient({}) }).should.throw(medium.MediumError) |
| done() |
| }) |
|  |
| it('should throw a MediumError when only clientId is provided', function (done) { |
| (function () { new medium.MediumClient({clientId: 'xxx'}) |
| }).should.throw(medium.MediumError) |
| done() |
| }) |
|  |
| it('should throw a MediumError when only clientSecret is provided', function (done) { |
| (function () { new medium.MediumClient({clientSecret: 'yyy'}) |
| }).should.throw(medium.MediumError) |
| done() |
| }) |
|  |
| it('should succeed when both clientId and clientSecret are provided', function (done) { |
| var client = new medium.MediumClient({clientId: 'xxx', clientSecret: 'yyy'}) |
| done() |
| }) |
| }) |
|  |
| describe('MediumClient - methods', function () { |

|  |
| --- |
|  |
| var clientId = 'xxx' |
| var clientSecret = 'yyy' |
| var client |
|  |
| beforeEach(function () { |
| client = new medium.MediumClient({clientId: clientId, clientSecret: clientSecret}) |
| nock.disableNetConnect() |
| }) |
|  |
| afterEach(function () { |
| nock.enableNetConnect(); |
| delete client |
| }) |
|  |
| describe('#setAccessToken', function () { |
|  |
| it ('sets the access token', function (done) { |
| var token = "new token" |
| client.setAccessToken(token) |
| client.\_accessToken.should.be.String().and.equal(token) |
| done() |
| }) |
| }) |
|  |
| describe('#getAuthorizationUrl', function () { |
|  |
| it ('returns a valid URL for fetching', function (done) { |
| var state = "state" |
| var redirectUrl = "https://example.com/callback" |
| var scope = [medium.Scope.BASIC\_PROFILE, medium.Scope.LIST\_PUBLICATIONS, |
| medium.Scope.PUBLISH\_POST] |
| var authUrlStr = client.getAuthorizationUrl(state, redirectUrl, scope) |
| var authUrl = url.parse(authUrlStr, true) |
| authUrl.protocol.should.equal('https:') |
| authUrl.hostname.should.equal('medium.com') |
| authUrl.pathname.should.equal('/m/oauth/authorize') |
| authUrl.query.should.deepEqual({ |
| client\_id: clientId, |
| scope: scope.join(','), |
| response\_type: 'code', |
| state: state, |
| redirect\_uri: redirectUrl |
| }) |
| done() |
| }) |
| }) |

|  |
| --- |
|  |
| describe('#exchangeAuthorizationCode', function () { |
|  |
| it ('makes a request for authorization\_code and sets the access token from response', |
| function (done) { |
| var code = '12345' |
| var grantType = 'authorization\_code' |
| var redirectUrl = 'https://example.com/callback' |
|  |
| var requestBody = qs.stringify({ |
| code: code, |
| client\_id: clientId, |
| client\_secret: clientSecret, |
| grant\_type: grantType, |
| redirect\_uri: redirectUrl |
| }) |
| // the response might have other parameters. this test only considers the ones called |
| out |
| // in the Medium Node SDK documentation |
| var accessToken = 'abcdef' |
| var refreshToken = 'ghijkl' |
| var responseBody = { |
| access\_token: accessToken, |
| refresh\_token: refreshToken |
| } |
| var request = nock('https://api.medium.com/', { |
| 'Content-Type': 'application/x-www-form-urlencoded' |
| }) |
| .post('/v1/tokens', requestBody) |
| .reply(201, responseBody) |
|  |
| client.exchangeAuthorizationCode(code, redirectUrl, function (err, data) { |
| if (err) throw err |
| data.access\_token.should.equal(accessToken) |
| data.refresh\_token.should.equal(refreshToken) |
| done() |
| }) |
| request.done() |
| }) |
| }) |
|  |
| describe('#exchangeRefreshToken', function () { |
|  |
| it ('makes a request for authorization\_code and sets the access token from response', |
| function (done) { |
| var refreshToken = 'fedcba' |
| var accessToken = 'lkjihg' |
|  |
| var requestBody = qs.stringify({ |
| refresh\_token: refreshToken, |
| client\_id: clientId, |
| client\_secret: clientSecret, |
| grant\_type: 'refresh\_token' |
| }) |
| // the response might have other parameters. this test only considers the ones called |
| out |
| // in the Medium Node SDK documentation |
| var responseBody = { |
| access\_token: accessToken, |
| refresh\_token: refreshToken |
| } |
| var request = nock('https://api.medium.com/', { |
| 'Content-Type': 'application/x-www-form-urlencoded' |
| }) |
| .post('/v1/tokens', requestBody) |
| .reply(201, responseBody) |
|  |
| client.exchangeRefreshToken(refreshToken, function (err, data) { |
| if (err) throw err |
| data.access\_token.should.equal(accessToken) |
| data.refresh\_token.should.equal(refreshToken) |
| done() |
| }) |
| request.done() |
| }) |
| }) |

1. Data Referencing Errors - None found.
2. Data Declaration Errors - None found.
3. Computation Errors - None found.
4. Comparison Errors

- The assertion client.\_accessToken.should.be.String().and.equal(token) in the setAccessToken test is checking if \_accessToken is a string before ensuring its equality. The order of assertions could lead to an unhandled error if \_accessToken is not defined or is not a string.

1. Control Flow Errors - None found.
2. Interface Errors

- The method client.setAccessToken(token) is called in the setAccessToken test, but if setAccessToken is not implemented correctly, it could lead to unexpected behavior.

7. Input/Output Errors

- The request.done() line in both exchangeAuthorizationCode and exchangeRefreshToken tests is incorrectly placed; it should be called after the request is executed, not directly after the request declaration. This may lead to premature invocation of done() in the context of network requests.

Next half of the code is here ::

|  |
| --- |
| describe('#getUser', function () { |
| it ('gets the information from expected URL and returns contents of data envelope', |
| function (done) { |
| var response = { data: 'response data' } |
|  |
| var request = nock('https://api.medium.com') |
| .get('/v1/me') |
| .reply(200, response) |
|  |
| client.getUser(function (err, data) { |
| if (err) throw err |
| data.should.deepEqual(response['data']) |
| done() |

|  |
| --- |
| }) |
| request.done() |
| }) |
| }) |
|  |
| describe('#getPublicationsForUser', function () { |
|  |
| it ('throws a MediumError when no user ID is provided', function (done) { |
| (function () { client.getPublicationsForUser({}) }).should.throw(medium.MediumError) |
| done() |
| }) |
|  |
| it ('makes a proper GET request to the Medium API and returns contents of data envelope |
| when valid options are provided', function (done) { |
| var userId = '123456' |
| var response = { data: 'response data' } |
|  |
| var request = nock('https://api.medium.com/') |
| .get('/v1/users/' + userId + '/publications') |
| .reply(200, response) |
|  |
| client.getPublicationsForUser({userId: userId}, function (err, data) { |
| if (err) throw err |
| data.should.deepEqual(response['data']) |
| done() |
| }) |
| request.done() |
| }) |
| }) |
|  |
| describe('#getContributorsForPublication', function () { |
|  |
| it ('throws a MediumError when no publication ID is provided', function (done) { |
| (function () { client.getContributorsForPublication({}) |
| }).should.throw(medium.MediumError) |
| done() |
| }) |
|  |
| it ('makes a proper GET request to the Medium API and returns contents of data envelope', |
| function (done) { |
| var options = { publicationId: 'abcdef' } |
| var response = { data: 'response data' } |
| var request = nock('https://api.medium.com/') |
| .get('/v1/publications/' + options.publicationId + '/contributors') |
| .reply(200, response) |
|  |
| client.getContributorsForPublication(options, function (err, data) { |

|  |
| --- |
| if (err) throw err |
| data.should.deepEqual(response['data']) |
| done() |
| }) |
| request.done() |
| }) |
| }) |
|  |
| describe('#createPost', function () { |
|  |
| it ('makes a proper POST request to the Medium API and returns contents of data envelope', |
| function (done) { |
| var options = { |
| userId: '123456', |
| title: 'new post title', |
| content: '<h1>New Post!</h1>', |
| contentFormat: 'html', |
| tags: ['js', 'unit tests'], |
| canonicalUrl: 'http://example.com/new-post', |
| publishedAt: '2004-02-12T15:19:21+00:00', |
| publishStatus: 'draft', |
| license: 'all-rights-reserved' |
| } |
| var response = { data: 'response data' } |
| var request = nock('https://api.medium.com/') |
| .post('/v1/users/' + options.userId + '/posts', { |
| title: options.title, |
| content: options.content, |
| contentFormat: options.contentFormat, |
| tags: options.tags, |
| canonicalUrl: options.canonicalUrl, |
| publishedAt: options.publishedAt, |
| publishStatus: options.publishStatus, |
| license: options.license |
| }) |
| .reply(200, response) |
|  |
| client.createPost(options, function (err, data) { |
| if (err) throw err |
| data.should.deepEqual(response['data']) |
| done() |
| }) |
| request.done() |
| }) |
| }) |
|  |
| describe('#createPostInPublication', function () { |

|  |
| --- |
|  |
| it ('should throw an error when no publication ID is provided', function (done) { |
| (function () { client.createPostInPublication({}) }).should.throw(medium.MediumError) |
| done() |
| }) |
|  |
| it ('makes a proper POST request to the Medium API and returns contents of data envelope', |
| function (done) { |
| var options = { |
| publicationId: 'abcdef', |
| title: 'new post title', |
| content: '<h1>New Post!</h1>', |
| contentFormat: 'html', |
| tags: ['js', 'unit tests'], |
| canonicalUrl: 'http://example.com/new-post', |
| publishedAt: '2004-02-12T15:19:21+00:00', |
| publishStatus: 'draft', |
| license: 'all-rights-reserved' |
| } |
| var response = { data: 'response data' } |
| var request = nock('https://api.medium.com/') |
| .post('/v1/publications/' + options.publicationId + '/posts', { |
| title: options.title, |
| content: options.content, |
| contentFormat: options.contentFormat, |
| tags: options.tags, |
| canonicalUrl: options.canonicalUrl, |
| publishedAt: options.publishedAt, |
| publishStatus: options.publishStatus, |
| license: options.license |
| }) |
| .reply(200, response) |
|  |
| client.createPostInPublication(options, function (err, data) { |
| if (err) throw err |
| data.should.deepEqual(response['data']) |
| done() |
| }) |
| request.done() |
| }) |
| }) |
| }) |

Here are the identified errors classified according to your categories in the provided code:

1. Data Referencing Errors - None found.
2. Data Declaration Errors - None found.
3. Computation Errors - None found.
4. Comparison Errors

- In the tests for getUser, getPublicationsForUser, getContributorsForPublication, createPost, and createPostInPublication, the assertion data.should.deepEqual(response['data']) assumes that response['data'] contains the expected output structure. If response.data is not defined correctly or is different in structure, this could lead to a comparison error.

1. Control Flow Errors - None found.
2. Interface Errors - None found.
3. Input/Output Errors

- The request.done() call in each test should be executed after the request is processed (after the callback), rather than immediately after the request declaration. This could lead to improper handling of request expectations.

**2nd Code is a sample of small Operating system**

https://github.com/nuta/operating-system-in-1000-lines/tree/main

There are sub parts to it so I placed all of them in random order…

|  |
| --- |
| #include "common.h" |
|  |
| void \*memset(void \*buf, char c, size\_t n) { |
| uint8\_t \*p = (uint8\_t \*) buf; |
| while (n--) |
| \*p++ = c; |
| return buf; |
| } |
|  |
| void \*memcpy(void \*dst, const void \*src, size\_t n) { |
| uint8\_t \*d = (uint8\_t \*) dst; |
| const uint8\_t \*s = (const uint8\_t \*) src; |
| while (n--) |
| \*d++ = \*s++; |
| return dst; |
| } |
|  |
| char \*strcpy(char \*dst, const char \*src) { |
| char \*d = dst; |
| while (\*src) |
| \*d++ = \*src++; |
| \*d = '\0'; |
| return dst; |
| } |
|  |
| int strcmp(const char \*s1, const char \*s2) { |
| while (\*s1 && \*s2) { |
| if (\*s1 != \*s2) |
| break; |

|  |
| --- |
| s1++; |
| s2++; |
| } |
|  |
| return \*(unsigned char \*)s1 - \*(unsigned char \*)s2; |
| } |
|  |
| void putchar(char ch); |
|  |
| void printf(const char \*fmt, ...) { |
| va\_list vargs; |
| va\_start(vargs, fmt); |
| while (\*fmt) { |
| if (\*fmt == '%') { |
| fmt++; |
| switch (\*fmt) { |
| case '\0': |
| putchar('%'); |
| goto end; |
| case '%': |
| putchar('%'); |
| break; |
| case 's': { |
| const char \*s = va\_arg(vargs, const char \*); |
| while (\*s) { |
| putchar(\*s); |
| s++; |
| } |
| break; |
| } |
| case 'd': { |
| int value = va\_arg(vargs, int); |
| if (value < 0) { |
| putchar('-'); |
| value = -value; |

|  |
| --- |
| } |
|  |
| int divisor = 1; |
| while (value / divisor > 9) |
| divisor \*= 10; |
|  |
| while (divisor > 0) { |
| putchar('0' + value / divisor); |
| value %= divisor; |
| divisor /= 10; |
| } |
|  |
| break; |
| } |
| case 'x': { |
| int value = va\_arg(vargs, int); |
| for (int i = 7; i >= 0; i--) { |
| int nibble = (value >> (i \* 4)) & 0xf; |
| putchar("0123456789abcdef"[nibble]); |
| } |
| } |
| } |
| } else { |
| putchar(\*fmt); |
| } |
|  |
| fmt++; |
| } |
|  |
| end: |
| va\_end(vargs); |
| } |

1. Data Referencing Errors - None found.
2. Data Declaration Errors

- In the printf function, the va\_list vargs is declared but not properly handled. If va\_end(vargs) is called without a corresponding va\_start(vargs, fmt), it could lead to undefined behavior, although this isn't directly indicated here since va\_start is correctly used before va\_end.

1. Computation Errors - None found.
2. Comparison Errors - None found.
3. Control Flow Errors

- In the printf function, the goto end; statement inside the switch block can create confusion. Although it is not an error, using goto can lead to less readable code and should be avoided if possible.

6. Interface Errors

- The putchar function is declared but not defined in the provided code. This could lead to linker errors if putchar is called without a definition available. - The function printf uses various formats (%d, %x, %s), but there is no error handling for unsupported formats, which could lead to unpredictable behavior if an unsupported format specifier is encountered.

7. Input/Output Errors

- In the printf function, there is no check for a null pointer in the const char \*s = va\_arg(vargs, const char \*); line for the string format specifier (%s). If a null pointer is passed, it could lead to dereferencing a null pointer and cause a segmentation fault.

|  |
| --- |
| #include "kernel.h" |
| #include "common.h" |
|  |
| extern char \_\_kernel\_base[]; |
| extern char \_\_stack\_top[]; |
| extern char \_\_bss[], \_\_bss\_end[]; |
| extern char \_\_free\_ram[], \_\_free\_ram\_end[]; |
| extern char \_binary\_shell\_bin\_start[], \_binary\_shell\_bin\_size[]; |
|  |
| struct process procs[PROCS\_MAX]; |
| struct process \*current\_proc; |
| struct process \*idle\_proc; |
|  |
| paddr\_t alloc\_pages(uint32\_t n) { |
| static paddr\_t next\_paddr = (paddr\_t) \_\_free\_ram; |
| paddr\_t paddr = next\_paddr; |
| next\_paddr += n \* PAGE\_SIZE; |
|  |
| if (next\_paddr > (paddr\_t) \_\_free\_ram\_end) |
| PANIC("out of memory"); |
|  |
| memset((void \*) paddr, 0, n \* PAGE\_SIZE); |
| return paddr; |
| } |
|  |
| void map\_page(uint32\_t \*table1, uint32\_t vaddr, paddr\_t paddr, uint32\_t flags) { |
| if (!is\_aligned(vaddr, PAGE\_SIZE)) |
| PANIC("unaligned vaddr %x", vaddr); |
|  |
| if (!is\_aligned(paddr, PAGE\_SIZE)) |
| PANIC("unaligned paddr %x", paddr); |
|  |
| uint32\_t vpn1 = (vaddr >> 22) & 0x3ff; |
| if ((table1[vpn1] & PAGE\_V) == 0) { |
| uint32\_t pt\_paddr = alloc\_pages(1); |
| table1[vpn1] = ((pt\_paddr / PAGE\_SIZE) << 10) | PAGE\_V; |
| } |
|  |

|  |
| --- |
| uint32\_t vpn0 = (vaddr >> 12) & 0x3ff; |
| uint32\_t \*table0 = (uint32\_t \*) ((table1[vpn1] >> 10) \* PAGE\_SIZE); |
| table0[vpn0] = ((paddr / PAGE\_SIZE) << 10) | flags | PAGE\_V; |
| } |
|  |
| struct sbiret sbi\_call(long arg0, long arg1, long arg2, long arg3, long arg4, |
| long arg5, long fid, long eid) { |
| register long a0 \_\_asm\_\_("a0") = arg0; |
| register long a1 \_\_asm\_\_("a1") = arg1; |
| register long a2 \_\_asm\_\_("a2") = arg2; |
| register long a3 \_\_asm\_\_("a3") = arg3; |
| register long a4 \_\_asm\_\_("a4") = arg4; |
| register long a5 \_\_asm\_\_("a5") = arg5; |
| register long a6 \_\_asm\_\_("a6") = fid; |
| register long a7 \_\_asm\_\_("a7") = eid; |
|  |
| \_\_asm\_\_ \_\_volatile\_\_("ecall" |
| : "=r"(a0), "=r"(a1) |
| : "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), |
| "r"(a6), "r"(a7) |
| : "memory"); |
| return (struct sbiret){.error = a0, .value = a1}; |
| } |
|  |
| struct virtio\_virtq \*blk\_request\_vq; |
| struct virtio\_blk\_req \*blk\_req; |
| paddr\_t blk\_req\_paddr; |
| unsigned blk\_capacity; |
|  |
| uint32\_t virtio\_reg\_read32(unsigned offset) { |
| return \*((volatile uint32\_t \*) (VIRTIO\_BLK\_PADDR + offset)); |
| } |
|  |
| uint64\_t virtio\_reg\_read64(unsigned offset) { |
| return \*((volatile uint64\_t \*) (VIRTIO\_BLK\_PADDR + offset)); |
| } |
|  |
| void virtio\_reg\_write32(unsigned offset, uint32\_t value) { |
| \*((volatile uint32\_t \*) (VIRTIO\_BLK\_PADDR + offset)) = value; |
| } |
|  |
| void virtio\_reg\_fetch\_and\_or32(unsigned offset, uint32\_t value) { |
| virtio\_reg\_write32(offset, virtio\_reg\_read32(offset) | value); |
| } |
|  |
| bool virtq\_is\_busy(struct virtio\_virtq \*vq) { |
| return vq->last\_used\_index != \*vq->used\_index; |

|  |
| --- |
| } |
|  |
| void virtq\_kick(struct virtio\_virtq \*vq, int desc\_index) { |
| vq->avail.ring[vq->avail.index % VIRTQ\_ENTRY\_NUM] = desc\_index; |
| vq->avail.index++; |
| \_\_sync\_synchronize(); |
| virtio\_reg\_write32(VIRTIO\_REG\_QUEUE\_NOTIFY, vq->queue\_index); |
| vq->last\_used\_index++; |
| } |
|  |
| struct virtio\_virtq \*virtq\_init(unsigned index) { |
| paddr\_t virtq\_paddr = alloc\_pages(align\_up(sizeof(struct virtio\_virtq), PAGE\_SIZE) / |
| PAGE\_SIZE); |
| struct virtio\_virtq \*vq = (struct virtio\_virtq \*) virtq\_paddr; |
| vq->queue\_index = index; |
| vq->used\_index = (volatile uint16\_t \*) &vq->used.index; |
| virtio\_reg\_write32(VIRTIO\_REG\_QUEUE\_SEL, index); |
| virtio\_reg\_write32(VIRTIO\_REG\_QUEUE\_NUM, VIRTQ\_ENTRY\_NUM); |
| virtio\_reg\_write32(VIRTIO\_REG\_QUEUE\_ALIGN, 0); |
| virtio\_reg\_write32(VIRTIO\_REG\_QUEUE\_PFN, virtq\_paddr); |
| return vq; |
| } |
|  |
| void virtio\_blk\_init(void) { |
| if (virtio\_reg\_read32(VIRTIO\_REG\_MAGIC) != 0x74726976) |
| PANIC("virtio: invalid magic value"); |
| if (virtio\_reg\_read32(VIRTIO\_REG\_VERSION) != 1) |
| PANIC("virtio: invalid version"); |
| if (virtio\_reg\_read32(VIRTIO\_REG\_DEVICE\_ID) != VIRTIO\_DEVICE\_BLK) |
| PANIC("virtio: invalid device id"); |
|  |
| virtio\_reg\_write32(VIRTIO\_REG\_DEVICE\_STATUS, 0); |
| virtio\_reg\_fetch\_and\_or32(VIRTIO\_REG\_DEVICE\_STATUS, VIRTIO\_STATUS\_ACK); |
| virtio\_reg\_fetch\_and\_or32(VIRTIO\_REG\_DEVICE\_STATUS, VIRTIO\_STATUS\_DRIVER); |
| virtio\_reg\_fetch\_and\_or32(VIRTIO\_REG\_DEVICE\_STATUS, VIRTIO\_STATUS\_FEAT\_OK); |
| blk\_request\_vq = virtq\_init(0); |
| virtio\_reg\_write32(VIRTIO\_REG\_DEVICE\_STATUS, VIRTIO\_STATUS\_DRIVER\_OK); |
|  |
| blk\_capacity = virtio\_reg\_read64(VIRTIO\_REG\_DEVICE\_CONFIG + 0) \* SECTOR\_SIZE; |
| printf("virtio-blk: capacity is %d bytes\n", blk\_capacity); |
|  |
| blk\_req\_paddr = alloc\_pages(align\_up(sizeof(\*blk\_req), PAGE\_SIZE) / PAGE\_SIZE); |
| blk\_req = (struct virtio\_blk\_req \*) blk\_req\_paddr; |
| } |

1. Data Referencing Errors - None identified.
2. Data Declaration Errors - None identified.
3. Computation Errors - None identified.
4. Comparison Errors - None identified.
5. Control Flow Errors

- No check for successful allocation in virtq\_init() after alloc\_pages(). This could lead to dereferencing a NULL pointer.

6. Interface Errors

- No explicit validation for register offsets in virtio\_reg\_read32, virtio\_reg\_read64, and related functions.

7. Input/Output Errors - None identified.

|  |
| --- |
| void read\_write\_disk(void \*buf, unsigned sector, int is\_write) { |
| if (sector >= blk\_capacity / SECTOR\_SIZE) { |
| printf("virtio: tried to read/write sector=%d, but capacity is %d\n", |
| sector, blk\_capacity / SECTOR\_SIZE); |
| return; |
| } |
|  |
| blk\_req->sector = sector; |
| blk\_req->type = is\_write ? VIRTIO\_BLK\_T\_OUT : VIRTIO\_BLK\_T\_IN; |
|  |
| if (is\_write) |
| memcpy(blk\_req->data, buf, SECTOR\_SIZE); |

|  |
| --- |
|  |
| struct virtio\_virtq \*vq = blk\_request\_vq; |
| vq->descs[0].addr = blk\_req\_paddr; |
| vq->descs[0].len = sizeof(uint32\_t) \* 2 + sizeof(uint64\_t); |
| vq->descs[0].flags = VIRTQ\_DESC\_F\_NEXT; |
| vq->descs[0].next = 1; |
|  |
| vq->descs[1].addr = blk\_req\_paddr + offsetof(struct virtio\_blk\_req, data); |
| vq->descs[1].len = SECTOR\_SIZE; |
| vq->descs[1].flags = VIRTQ\_DESC\_F\_NEXT | (is\_write ? 0 : VIRTQ\_DESC\_F\_WRITE); |
| vq->descs[1].next = 2; |
|  |
| vq->descs[2].addr = blk\_req\_paddr + offsetof(struct virtio\_blk\_req, status); |
| vq->descs[2].len = sizeof(uint8\_t); |
| vq->descs[2].flags = VIRTQ\_DESC\_F\_WRITE; |
|  |
| virtq\_kick(vq, 0); |
| while (virtq\_is\_busy(vq)) |
| ; |
|  |
| if (blk\_req->status != 0) { |
| printf("virtio: warn: failed to read/write sector=%d status=%d\n", |
| sector, blk\_req->status); |
| return; |
| } |
|  |
| if (!is\_write) |
| memcpy(buf, blk\_req->data, SECTOR\_SIZE); |
| } |
|  |
| struct file files[FILES\_MAX]; |
| uint8\_t disk[DISK\_MAX\_SIZE]; |
|  |
| int oct2int(char \*oct, int len) { |
| int dec = 0; |
| for (int i = 0; i < len; i++) { |
| if (oct[i] < '0' || oct[i] > '7') |
| break; |
|  |
| dec = dec \* 8 + (oct[i] - '0'); |
| } |
| return dec; |
| } |
|  |
| void fs\_flush(void) { |
| memset(disk, 0, sizeof(disk)); |
| unsigned off = 0; |

|  |
| --- |
| for (int file\_i = 0; file\_i < FILES\_MAX; file\_i++) { |
| struct file \*file = &files[file\_i]; |
| if (!file->in\_use) |
| continue; |
|  |
| struct tar\_header \*header = (struct tar\_header \*) &disk[off]; |
| memset(header, 0, sizeof(\*header)); |
| strcpy(header->name, file->name); |
| strcpy(header->mode, "000644"); |
| strcpy(header->magic, "ustar"); |
| strcpy(header->version, "00"); |
| header->type = '0'; |
|  |
| int filesz = file->size; |
| for (int i = sizeof(header->size); i > 0; i--) { |
| header->size[i - 1] = (filesz % 8) + '0'; |
| filesz /= 8; |
| } |
|  |
| int checksum = ' ' \* sizeof(header->checksum); |
| for (unsigned i = 0; i < sizeof(struct tar\_header); i++) |
| checksum += (unsigned char) disk[off + i]; |
|  |
| for (int i = 5; i >= 0; i--) { |
| header->checksum[i] = (checksum % 8) + '0'; |
| checksum /= 8; |
| } |
|  |
| memcpy(header->data, file->data, file->size); |
| off += align\_up(sizeof(struct tar\_header) + file->size, SECTOR\_SIZE); |
| } |
|  |
| for (unsigned sector = 0; sector < sizeof(disk) / SECTOR\_SIZE; sector++) |
| read\_write\_disk(&disk[sector \* SECTOR\_SIZE], sector, true); |
|  |
| printf("wrote %d bytes to disk\n", sizeof(disk)); |
| } |
|  |
| void fs\_init(void) { |
| for (unsigned sector = 0; sector < sizeof(disk) / SECTOR\_SIZE; sector++) |
| read\_write\_disk(&disk[sector \* SECTOR\_SIZE], sector, false); |
|  |
| unsigned off = 0; |
| for (int i = 0; i < FILES\_MAX; i++) { |
| struct tar\_header \*header = (struct tar\_header \*) &disk[off]; |
| if (header->name[0] == '\0') |
| break; |
|  |
| if (strcmp(header->magic, "ustar") != 0) |
| PANIC("invalid tar header: magic=\"%s\"", header->magic); |
|  |
| int filesz = oct2int(header->size, sizeof(header->size)); |
| struct file \*file = &files[i]; |
| file->in\_use = true; |
| strcpy(file->name, header->name); |
| memcpy(file->data, header->data, filesz); |
| file->size = filesz; |
| printf("file: %s, size=%d\n", file->name, file->size); |
|  |
| off += align\_up(sizeof(struct tar\_header) + filesz, SECTOR\_SIZE); |
| } |
| } |
|  |

1. Data Referencing Errors

- The code references blk\_req, blk\_capacity, blk\_request\_vq, and blk\_req\_paddr without showing their definitions. Make sure these variables are properly initialized and referenced.

2. Data Declaration Errors

* The variable disk is declared with uint8\_t disk[DISK\_MAX\_SIZE];, but there’s no indication of the value assigned to DISK\_MAX\_SIZE. Ensure it’s defined somewhere.
* The struct tar\_header is referenced without a declaration in the provided code. Ensure it is defined correctly in your project.

3. Computation Errors

* The calculation of filesz in fs\_flush does not account for the potential overflow when calculating the checksum. Although the tar format specifies a maximum size, it’s a good practice to check sizes to avoid overflow.
* In the oct2int function, if the input oct string has more than three characters (which represent a valid octal digit), the conversion might give unexpected results. Consider adding a limit on len.

4. Comparison Errors

- In fs\_init, the check if (strcmp(header->magic, "ustar") != 0) is valid, but the code doesn't handle the case where header->magic could be NULL. Consider adding a NULL check before comparison.

5. Control Flow Errors

* The read\_write\_disk function might enter an infinite loop if the disk request is never completed. Ensure that virtq\_kick(vq, 0) and virtq\_is\_busy(vq) are implemented correctly to handle this situation.
* The function fs\_flush will print that it has written to the disk regardless of whether the write was successful. Consider checking for errors in read\_write\_disk.

6. Interface Errors

* The putchar function must be defined elsewhere, or else there will be linking errors when compiling.
* Ensure that align\_up is properly defined and that its purpose is clear; it seems to be intended for aligning data sizes, but its implementation is not provided here.

7. Input/Output Errors

* In the fs\_flush function, when writing to the disk, if read\_write\_disk fails for any reason (e.g., due to a full disk or hardware failure), the user is not notified.

Implement error handling to manage this.

* In the fs\_init function, if the data read from the disk doesn’t match the expected format or the file size exceeds DISK\_MAX\_SIZE, it may cause out-of-bounds memory access when populating the file structures.

|  |
| --- |
| struct file \*fs\_lookup(const char \*filename) { |
| for (int i = 0; i < FILES\_MAX; i++) { |
| struct file \*file = &files[i]; |
| if (!strcmp(file->name, filename)) |
| return file; |
| } |
|  |
| return NULL; |
| } |
|  |
| void putchar(char ch) { |
| sbi\_call(ch, 0, 0, 0, 0, 0, 0, 1 /\* Console Putchar \*/); |
| } |
|  |
| long getchar(void) { |
| struct sbiret ret = sbi\_call(0, 0, 0, 0, 0, 0, 0, 2); |
| return ret.error; |
| } |
|  |
| \_\_attribute\_\_((naked)) |
| \_\_attribute\_\_((aligned(4))) |
| void kernel\_entry(void) { |
| \_\_asm\_\_ \_\_volatile\_\_( |
| "csrrw sp, sscratch, sp\n" |
| "addi sp, sp, -4 \* 31\n" |
| "sw ra, 4 \* 0(sp)\n" |
| "sw gp, 4 \* 1(sp)\n" |
| "sw tp, 4 \* 2(sp)\n" |
| "sw t0, 4 \* 3(sp)\n" |
| "sw t1, 4 \* 4(sp)\n" |
| "sw t2, 4 \* 5(sp)\n" |
| "sw t3, 4 \* 6(sp)\n" |
| "sw t4, 4 \* 7(sp)\n" |
| "sw t5, 4 \* 8(sp)\n" |
| "sw t6, 4 \* 9(sp)\n" |
| "sw a0, 4 \* 10(sp)\n" |
| "sw a1, 4 \* 11(sp)\n" |
| "sw a2, 4 \* 12(sp)\n" |
| "sw a3, 4 \* 13(sp)\n" |
| "sw a4, 4 \* 14(sp)\n" |
| "sw a5, 4 \* 15(sp)\n" |

|  |
| --- |
| "sw a6, 4 \* 16(sp)\n" |
| "sw a7, 4 \* 17(sp)\n" |
| "sw s0, 4 \* 18(sp)\n" |
| "sw s1, 4 \* 19(sp)\n" |
| "sw s2, 4 \* 20(sp)\n" |
| "sw s3, 4 \* 21(sp)\n" |
| "sw s4, 4 \* 22(sp)\n" |
| "sw s5, 4 \* 23(sp)\n" |
| "sw s6, 4 \* 24(sp)\n" |
| "sw s7, 4 \* 25(sp)\n" |
| "sw s8, 4 \* 26(sp)\n" |
| "sw s9, 4 \* 27(sp)\n" |
| "sw s10, 4 \* 28(sp)\n" |
| "sw s11, 4 \* 29(sp)\n" |
|  |
| "csrr a0, sscratch\n" |
| "sw a0, 4 \* 30(sp)\n" |
|  |
| "addi a0, sp, 4 \* 31\n" |
| "csrw sscratch, a0\n" |
|  |
| "mv a0, sp\n" |
| "call handle\_trap\n" |
|  |
| "lw ra, 4 \* 0(sp)\n" |
| "lw gp, 4 \* 1(sp)\n" |
| "lw tp, 4 \* 2(sp)\n" |
| "lw t0, 4 \* 3(sp)\n" |
| "lw t1, 4 \* 4(sp)\n" |
| "lw t2, 4 \* 5(sp)\n" |
| "lw t3, 4 \* 6(sp)\n" |
| "lw t4, 4 \* 7(sp)\n" |
| "lw t5, 4 \* 8(sp)\n" |
| "lw t6, 4 \* 9(sp)\n" |
| "lw a0, 4 \* 10(sp)\n" |
| "lw a1, 4 \* 11(sp)\n" |
| "lw a2, 4 \* 12(sp)\n" |
| "lw a3, 4 \* 13(sp)\n" |
| "lw a4, 4 \* 14(sp)\n" |
| "lw a5, 4 \* 15(sp)\n" |
| "lw a6, 4 \* 16(sp)\n" |
| "lw a7, 4 \* 17(sp)\n" |
| "lw s0, 4 \* 18(sp)\n" |
| "lw s1, 4 \* 19(sp)\n" |
| "lw s2, 4 \* 20(sp)\n" |
| "lw s3, 4 \* 21(sp)\n" |
| "lw s4, 4 \* 22(sp)\n" |

|  |
| --- |
| "lw s5, 4 \* 23(sp)\n" |
| "lw s6, 4 \* 24(sp)\n" |
| "lw s7, 4 \* 25(sp)\n" |
| "lw s8, 4 \* 26(sp)\n" |
| "lw s9, 4 \* 27(sp)\n" |
| "lw s10, 4 \* 28(sp)\n" |
| "lw s11, 4 \* 29(sp)\n" |
| "lw sp, 4 \* 30(sp)\n" |
| "sret\n" |
| ); |
| } |
|  |
| \_\_attribute\_\_((naked)) void user\_entry(void) { |
| \_\_asm\_\_ \_\_volatile\_\_( |
| "csrw sepc, %[sepc]\n" |
| "csrw sstatus, %[sstatus]\n" |
| "sret\n" |
| : |
| : [sepc] "r" (USER\_BASE), |
| [sstatus] "r" (SSTATUS\_SPIE | SSTATUS\_SUM) |
| ); |
| } |
|  |
| \_\_attribute\_\_((naked)) void switch\_context(uint32\_t \*prev\_sp, |
| uint32\_t \*next\_sp) { |
| \_\_asm\_\_ \_\_volatile\_\_( |
| "addi sp, sp, -13 \* 4\n" |
| "sw ra, 0 \* 4(sp)\n" |
| "sw s0, 1 \* 4(sp)\n" |
| "sw s1, 2 \* 4(sp)\n" |
| "sw s2, 3 \* 4(sp)\n" |
| "sw s3, 4 \* 4(sp)\n" |
| "sw s4, 5 \* 4(sp)\n" |
| "sw s5, 6 \* 4(sp)\n" |
| "sw s6, 7 \* 4(sp)\n" |
| "sw s7, 8 \* 4(sp)\n" |
| "sw s8, 9 \* 4(sp)\n" |
| "sw s9, 10 \* 4(sp)\n" |
| "sw s10, 11 \* 4(sp)\n" |
| "sw s11, 12 \* 4(sp)\n" |
| "sw sp, (a0)\n" |
| "lw sp, (a1)\n" |
| "lw ra, 0 \* 4(sp)\n" |
| "lw s0, 1 \* 4(sp)\n" |
| "lw s1, 2 \* 4(sp)\n" |
| "lw s2, 3 \* 4(sp)\n" |
| "lw s3, 4 \* 4(sp)\n" |

|  |
| --- |
| "lw s4, 5 \* 4(sp)\n" |
| "lw s5, 6 \* 4(sp)\n" |
| "lw s6, 7 \* 4(sp)\n" |
| "lw s7, 8 \* 4(sp)\n" |
| "lw s8, 9 \* 4(sp)\n" |
| "lw s9, 10 \* 4(sp)\n" |
| "lw s10, 11 \* 4(sp)\n" |
| "lw s11, 12 \* 4(sp)\n" |
| "addi sp, sp, 13 \* 4\n" |
| "ret\n" |
| ); |
| } |
|  |
| struct process \*create\_process(const void \*image, size\_t image\_size) { |
| struct process \*proc = NULL; |
| int i; |
| for (i = 0; i < PROCS\_MAX; i++) { |
| if (procs[i].state == PROC\_UNUSED) { |
| proc = &procs[i]; |
| break; |
| } |
| } |
|  |
| if (!proc) |
| PANIC("no free process slots"); |
|  |
| uint32\_t \*sp = (uint32\_t \*) &proc->stack[sizeof(proc->stack)]; |
| \*--sp = 0; // s11 |
| \*--sp = 0; // s10 |
| \*--sp = 0; // s9 |
| \*--sp = 0; // s8 |
| \*--sp = 0; // s7 |
| \*--sp = 0; // s6 |
| \*--sp = 0; // s5 |
| \*--sp = 0; // s4 |
| \*--sp = 0; // s3 |
| \*--sp = 0; // s2 |
| \*--sp = 0; // s1 |
| \*--sp = 0; // s0 |
| \*--sp = (uint32\_t) user\_entry; // ra |
|  |
| uint32\_t \*page\_table = (uint32\_t \*) alloc\_pages(1); |
|  |
| // Kernel pages. |
| for (paddr\_t paddr = (paddr\_t) \_\_kernel\_base; |
| paddr < (paddr\_t) \_\_free\_ram\_end; paddr += PAGE\_SIZE) |
| map\_page(page\_table, paddr, paddr, PAGE\_R | PAGE\_W | PAGE\_X); |
|  |
| // virtio-blk |
| map\_page(page\_table, VIRTIO\_BLK\_PADDR, VIRTIO\_BLK\_PADDR, PAGE\_R | PAGE\_W); |
|  |
| // User pages. |
| for (uint32\_t off = 0; off < image\_size; off += PAGE\_SIZE) { |
| paddr\_t page = alloc\_pages(1); |
| memcpy((void \*) page, image + off, PAGE\_SIZE); |
| map\_page(page\_table, USER\_BASE + off, page, |
| PAGE\_U | PAGE\_R | PAGE\_W | PAGE\_X); |
| } |
|  |
| proc->pid = i + 1; |
| proc->state = PROC\_RUNNABLE; |
| proc->sp = (uint32\_t) sp; |
| proc->page\_table = page\_table; |
| return proc; |
| } |

1. Data Referencing Errors

- Potential Null Pointer Dereference: The fs\_lookup function assumes that files is initialized and valid. If files is uninitialized or if FILES\_MAX is set to 0, it may lead to undefined behavior.

2. Data Declaration Errors

- Missing Struct Definition: The struct file and the files array are referenced but not defined in the provided code. This could lead to compilation errors if they are not declared elsewhere in the program.

1. Computation Errors - None found.
2. Comparison Errors - None found.
3. Control Flow Errors

- Unconditional Exit: The PANIC("no free process slots"); call does not handle the case where proc is NULL gracefully, potentially leading to abrupt termination of the program. Instead, it should ideally return or clean up resources.

1. Interface Errors - None found.
2. Input/Output Errors

* Buffer Overrun Risk: The loop that initializes the stack (with \*--sp = 0;) assumes that the stack has sufficient space. If the size of proc->stack is less than expected, it may result in a stack overflow.

|  |
| --- |
| void yield(void) { |
| struct process \*next = idle\_proc; |
| for (int i = 0; i < PROCS\_MAX; i++) { |
| struct process \*proc = &procs[(current\_proc->pid + i) % PROCS\_MAX]; |
| if (proc->state == PROC\_RUNNABLE && proc->pid > 0) { |
| next = proc; |
| break; |
| } |
| } |
|  |
| if (next == current\_proc) |
| return; |
|  |
| struct process \*prev = current\_proc; |
| current\_proc = next; |
|  |
| \_\_asm\_\_ \_\_volatile\_\_( |
| "sfence.vma\n" |
| "csrw satp, %[satp]\n" |
| "sfence.vma\n" |
| "csrw sscratch, %[sscratch]\n" |
| : |
| : [satp] "r" (SATP\_SV32 | ((uint32\_t) next->page\_table / PAGE\_SIZE)), |
| [sscratch] "r" ((uint32\_t) &next->stack[sizeof(next->stack)]) |
| ); |
|  |
| switch\_context(&prev->sp, &next->sp); |
| } |

|  |
| --- |
|  |
| void handle\_syscall(struct trap\_frame \*f) { |
| switch (f->a3) { |
| case SYS\_PUTCHAR: |
| putchar(f->a0); |
| break; |
| case SYS\_GETCHAR: |
| while (1) { |
| long ch = getchar(); |
| if (ch >= 0) { |
| f->a0 = ch; |
| break; |
| } |
|  |
| yield(); |
| } |
| break; |
| case SYS\_EXIT: |
| printf("process %d exited\n", current\_proc->pid); |
| current\_proc->state = PROC\_EXITED; |
| yield(); |
| PANIC("unreachable"); |
| case SYS\_READFILE: |
| case SYS\_WRITEFILE: { |
| const char \*filename = (const char \*) f->a0; |
| char \*buf = (char \*) f->a1; |
| int len = f->a2; |
| struct file \*file = fs\_lookup(filename); |
| if (!file) { |
| printf("file not found: %s\n", filename); |
| f->a0 = -1; |
| break; |
| } |
|  |
| if (len > (int) sizeof(file->data)) |
| len = file->size; |
|  |
| if (f->a3 == SYS\_WRITEFILE) { |
| memcpy(file->data, buf, len); |
| file->size = len; |
| fs\_flush(); |
| } else { |
| memcpy(buf, file->data, len); |
| } |
|  |
| f->a0 = len; |
| break; |

|  |
| --- |
| } |
| default: |
| PANIC("unexpected syscall a3=%x\n", f->a3); |
| } |
| } |
|  |
| void handle\_trap(struct trap\_frame \*f) { |
| uint32\_t scause = READ\_CSR(scause); |
| uint32\_t stval = READ\_CSR(stval); |
| uint32\_t user\_pc = READ\_CSR(sepc); |
| if (scause == SCAUSE\_ECALL) { |
| handle\_syscall(f); |
| user\_pc += 4; |
| } else { |
| PANIC("unexpected trap scause=%x, stval=%x, sepc=%x\n", scause, stval, user\_pc); |
| } |
|  |
| WRITE\_CSR(sepc, user\_pc); |
| } |
|  |
| void kernel\_main(void) { |
| memset(\_\_bss, 0, (size\_t) \_\_bss\_end - (size\_t) \_\_bss); |
| printf("\n\n"); |
| WRITE\_CSR(stvec, (uint32\_t) kernel\_entry); |
| virtio\_blk\_init(); |
| fs\_init(); |
|  |
| idle\_proc = create\_process(NULL, 0); |
| idle\_proc->pid = -1; // idle |
| current\_proc = idle\_proc; |
|  |
| create\_process(\_binary\_shell\_bin\_start, (size\_t) \_binary\_shell\_bin\_size); |
| yield(); |
|  |
| PANIC("switched to idle process"); |
| } |
|  |
| \_\_attribute\_\_((section(".text.boot"))) |
| \_\_attribute\_\_((naked)) |
| void boot(void) { |
| \_\_asm\_\_ \_\_volatile\_\_( |
| "mv sp, %[stack\_top]\n" |
| "j kernel\_main\n" |
| : |
| : [stack\_top] "r" (\_\_stack\_top) |
| ); |
| } |

Data Referencing Errors

* Potential Null Pointer Dereference: current\_proc could be null if no processes have been created or if it has been improperly initialized before yield() is called.

Data Declaration Errors

* Uninitialized Variables: Variables such as idle\_proc and current\_proc may be used without proper initialization if create\_process fails or if there are no processes.

Computation Errors

* Improper Memory Access: The calculation of next->page\_table / PAGE\_SIZE could lead to incorrect values if next->page\_table is not properly aligned or initialized.

Comparison Errors

* Unsigned vs. Signed Comparison: Comparing proc->pid > 0 may cause unintended behavior if proc->pid is an unsigned type.

Control Flow Errors

* Infinite Loop Risk: The while (1) loop in handle\_syscall for SYS\_GETCHAR may lead to an infinite loop if getchar() never returns a valid character.

Interface Errors

* Missing Error Handling for System Calls: Functions like fs\_lookup, memcpy, and printf may fail silently without error checking or reporting in certain scenarios.

Input/Output Errors

* Data Overwrite Risk: In handle\_syscall for SYS\_WRITEFILE, if len is not properly validated, it may lead to writing beyond the bounds of file->data.

|  |
| --- |
| #include "user.h" |
|  |
| void main(void) { |
| while (1) { |
| prompt: |
| printf("> "); |
| char cmdline[128]; |
| for (int i = 0;; i++) { |
| char ch = getchar(); |
| putchar(ch); |
| if (i == sizeof(cmdline) - 1) { |
| printf("command line too long\n"); |
| goto prompt; |
| } else if (ch == '\r') { |
| printf("\n"); |
| cmdline[i] = '\0'; |
| break; |
| } else { |
| cmdline[i] = ch; |
| } |
| } |
|  |
| if (strcmp(cmdline, "hello") == 0) |
| printf("Hello world from shell!\n"); |
| else if (strcmp(cmdline, "exit") == 0) |
| exit(); |
| else if (strcmp(cmdline, "readfile") == 0) { |
| char buf[128]; |
| int len = readfile("hello.txt", buf, sizeof(buf)); |
| buf[len] = '\0'; |
| printf("%s\n", buf); |
| } |
| else if (strcmp(cmdline, "writefile") == 0) |
| writefile("hello.txt", "Hello from shell!\n", 19); |
| else |
| printf("unknown command: %s\n", cmdline); |
| } |
| } |

Data Referencing Errors

* Potential Buffer Overflow: The cmdline buffer is not properly null-terminated if the user inputs more than 127 characters (since one byte is used for the null terminator).

Data Declaration Errors

* Uninitialized Variable: The variable buf in the readfile command could be uninitialized if the file reading fails before it is populated.

Computation Errors

* Length Calculation: In the readfile command, the length returned by readfile() is used directly without checking if it exceeds the size of buf. If len is larger than 128, this could lead to a buffer overflow when setting buf[len] =

'\0';.

Comparison Errors

* Use of strcmp: If cmdline is not properly null-terminated due to buffer overflow or a missed termination case, the behavior of strcmp can be undefined.

Control Flow Errors

* Infinite Loop Risk: The while (1) loop will run indefinitely unless a command that calls exit() is executed. There's no condition to break out of the loop except for exit().

Interface Errors

* Missing Error Handling: The return value of readfile is not checked for errors.

If the file does not exist or read fails, it could lead to undefined behavior.

Input/Output Errors

* Data Overwrite Risk: In the writefile command, there is no check to ensure that the data being written is less than or equal to the length of the file buffer on the file system.

|  |
| --- |
| #include "user.h" |
|  |
| extern char \_\_stack\_top[]; |
| int syscall(int sysno, int arg0, int arg1, int arg2) { |
| register int a0 \_\_asm\_\_("a0") = arg0; |
| register int a1 \_\_asm\_\_("a1") = arg1; |
| register int a2 \_\_asm\_\_("a2") = arg2; register int a3 \_\_asm\_\_("a3") = sysno; |
|  |
| \_\_asm\_\_ \_\_volatile\_\_("ecall" |
| : "=r"(a0)  : "r"(a0), "r"(a1), "r"(a2), "r"(a3) |
| : "memory"); |
|  |
| return a0;  } |
|  |
| void putchar(char ch) { |
| syscall(SYS\_PUTCHAR, ch, 0, 0);  } |
|  |
| int getchar(void) { |
| return syscall(SYS\_GETCHAR, 0, 0, 0);  } |
|  |
| int readfile(const char \*filename, char \*buf, int len) { |
| return syscall(SYS\_READFILE, (int) filename, (int) buf, len);  } |
|  |
| int writefile(const char \*filename, const char \*buf, int len) { |
| return syscall(SYS\_WRITEFILE, (int) filename, (int) buf, len); |
| } |
| \_\_attribute\_\_((noreturn)) void exit(void) { |
| syscall(SYS\_EXIT, 0, 0, 0); |
| for (;;); |
| } |
| \_\_attribute\_\_((section(".text.start"))) |
| \_\_attribute\_\_((naked)) |
| void start(void) {  \_\_asm\_\_ \_\_volatile\_\_( |
| "mv sp, %[stack\_top]\n" |
| "call main\n" |
| "call exit\n" ::[stack\_top] "r"(\_\_stack\_top));  } |

Data Referencing Errors

* Casting Pointers to Integers: The code casts const char \*filename and char \*buf to int, which can lead to data loss or corruption on architectures where pointers are larger than integers (e.g., 64-bit systems).

Data Declaration Errors

* Uninitialized Variables: If syscall fails or returns an error value, the variables buf and filename may not be handled properly in readfile and writefile functions, which could lead to unexpected behavior.

Computation Errors

* Return Value Ignored: In readfile and writefile, the return value from syscall is not checked. If the syscall fails (e.g., file not found), this could lead to undefined behavior when using the data later.

Comparison Errors

* No apparent comparison errors exist in the provided code.

Control Flow Errors

* Endless Loop in exit: The for (;;); loop in the exit function will create an infinite loop after the syscall call, which could indicate a lack of proper termination or error handling.

Interface Errors

* No Error Handling for System Calls: There is no error checking for the return values of syscall in any function. For instance, if a file operation fails, the error is not handled.

Input/Output Errors

* Invalid Memory Access: If buf in readfile or writefile points to an invalid or unallocated memory address, the code will attempt to read from or write to that memory location, leading to potential crashes or data corruption.