3)

SEND\_CHANNEL:

A channel is simply a FIFO stack/queue in memory that multiple processes can access. When we pop the channel from the stack, we are receiving 2 values, the address of the beginning of the stack, and the length of the stack, or the last index of the stack. Using a while loop, which branches back to a particular bytecode when a condition is met, we can repeatedly poll our FIFO stack if our channel or stack is full.

RECV\_CHANNEL:

This also operates similarly to SEND\_CHANNEL in that it polls a particular FIFO stack/queue in memory. Difference here being that once it gets a value from the channel, it doesn’t block, and pushes that value onto the stack.

SPAWN:

To spawn 2 separate functions concurrently, we need to be able to save the current state of each operating function before switching to the next. This is also known as context switching, where the state of the current operating function/thread must be saved before switching to a different operating function/thread, and the new state of the new function must be restored onto the stack. So, in our interpreter, the state of the function (variables assigned) is already being written to memory using WRITE\_VAR.

4) Solution in find\_files.sh. Run the script, then command is executed ‘find\_files {FILE\_EXTENSION}” where {FILE\_EXTENSION} is replaced with the file extension you want to recursively find. This will also output the number of lines in each file

A picture containing text, clock, dark

Description automatically generated

5)

Hashing provides deterministic output, preimage resistance, randomized, and is often collision

resistant. An important feature of hashing is that it’s easy to verify the key of a particular result of a hash, but difficult to find the input to that hash result. The most notable way hashing is used in PoW blockchains where miners try to

compete to find a valid nonce to help form consensus on the blockchain. Once one miner finds

the valid nonce to the hash, they can propagate that result to the network, where other miners

can quickly and efficiently verify the result.

Hashing also one the underlying tools behind public/private key cryptography. They are used to

help users sign transactions with their private key. Others can than verify the transaction using

their public key.

Another notable storage technique used in blockchains are Merkle trees, which are created by

combining leaves in pairs, and hashing those pairs. This process is repeated on the results again

by combining the results in pairs, and hashing those pairs. This is done till a single result is

found. Merkle trees can be used to help verify data that is sent across a blockchain network is

unaltered and not corrupted. For example in lightweight clients, clients might not send over all the transactions to be included, but instead only the merkle root of those transactions, which they then can verify using a merkle proof.

6) Bitcoins UTXO model is a model where users on the network have a set of UTXOs for users, or unspent transaction outputs. These unspent transactions determine the amount of bitcoin a user can send. Once a user sends some bitcoin to another user on the network, 2 new UTXOs are created, 1 UTXO to a new user, and 1 to themselves. Adding up your UTXOs allows a user to see how many unspent bitcoin transactions were made to them, which they could spend.

7) Bitcoin’s block structure contains 5 things: a magic number, block size, block header,

transaction counter, and the transactions included in the block. Each block has a reference to the processed block before it, creating a chain of blocks each referencing a block of older confirmed transactions before it.

8) POW/POS are both consensus mechanisms used to verify new transactions and put them on the blockchain. They both are trying to solve the double spending issue where the same token can be spent more than once. They are both trying to solve the problem of how to incentivize miners/validators from acting maliciously and putting in fake transactions into the block. PoW does this by making it computationally expensive to produce a hash of a block with malicious transactions. If an attacker attempts to create a block with a. malicious transaction, they will have to spend significant computational resources to find the correct nonce to the block. PoS does this by having validators staking some crypto, and if any fraudulent activity is detected, the validator loses all their stake. The idea is that the validator’s stake is larger than the fees earned from the block, disincentivizing the validator from putting in malicious transactions.