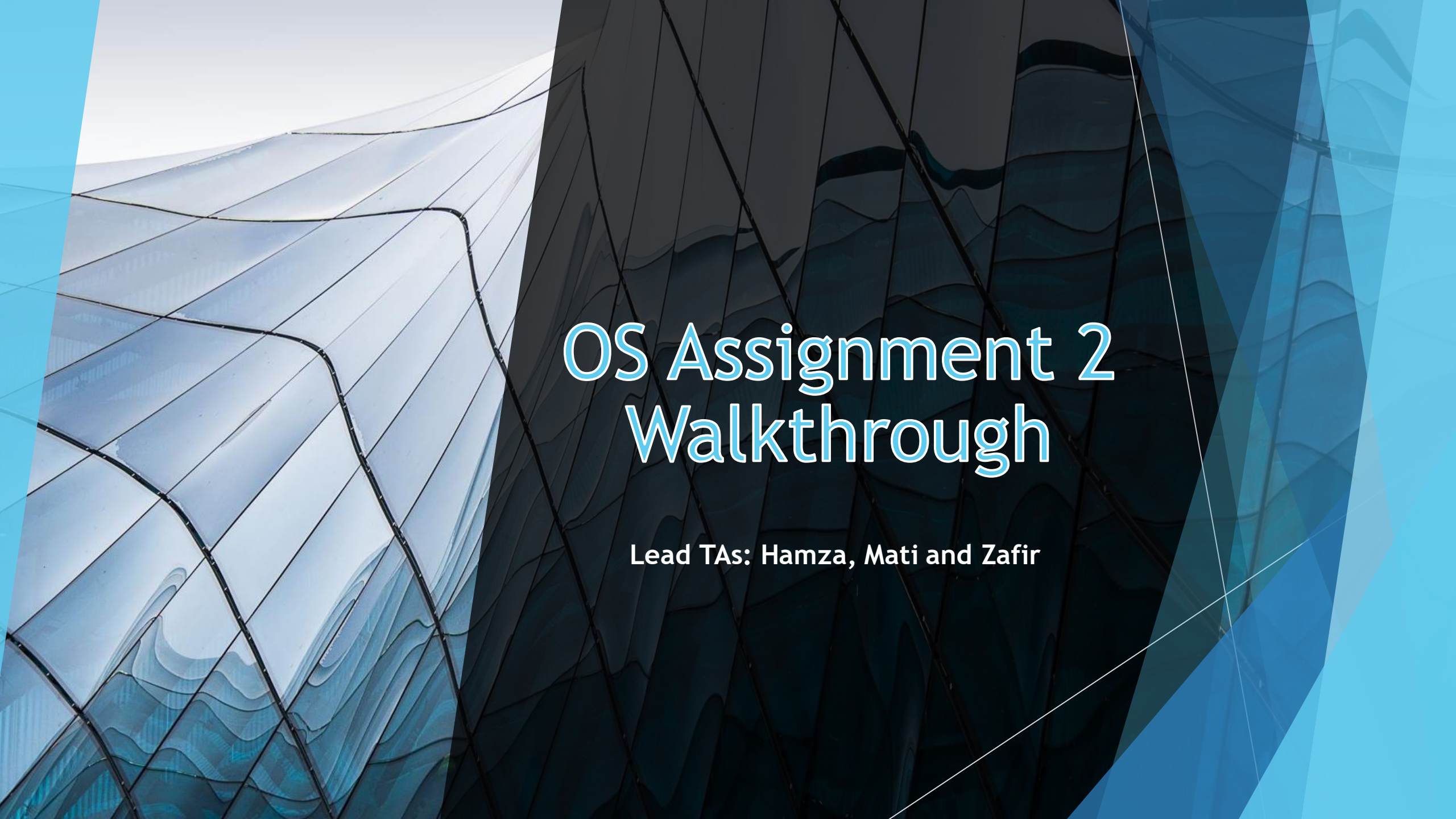


OS Assignment 2 Walkthrough

Lead TAs: Hamza, Mati and Zafir

What is Shell

- A shell is a computer program that exposes the operating system's services to a human user or some other program.
- Using the shell program, a user can use the operating system services to execute several different processes.
- For example, One of the most common linux commands executed via the shell is *ls*, which is used to list all the files, directories and processes within a particular directory
- In this assignment, you will be creating a simple shell program that will perform a subset of the functionalities of the original shell.

System Calls

fork(): Creates a new process by duplicating the calling process. The new process is the child process, the calling process is called the parent process

exec(): Used to replace the current process image with a new process image i.e, a new process will now replace the current one and start executing

wait(): Used for blocking the parent process from executing till a child process of the parent has terminated or a signal is received by the parent from a child.

dup2(oldfd,newfd): Makes the newfd file descriptor refer to the same file as the oldfd file descriptor.

pipe(): Used for interprocess communication such that the output of one process acts as the input of another process

Inter-Process Communication (Revision)

- Used by two processes to communicate with eachother
- There is an in memory queue to which one of the processes writes and the other reads from it.
- The writing process is called producer and the reading process is called consumer
- Two separate file descriptors are used for reading and writing each.

Environment Setup

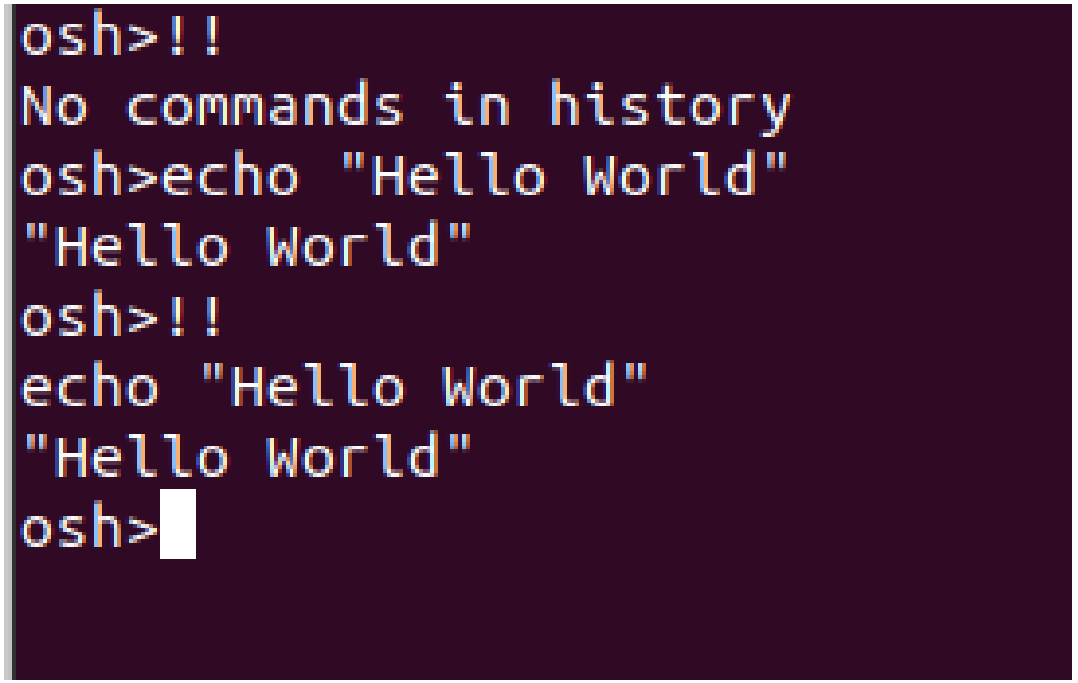
- This assignment does not require any specific environment and can be completed on any Unix Based system that support system calls.

Part 1

- Fork a child process to execute the command specified by the user.
- Only the first word of the string entered by the user is the actual command. Rest of the words are arguments to that command.
- This will require string parsing to separate the first word from the rest of the word.
- After string parsing,
  pass the first word as the first command to the *execvp* system call, and the entire user entered string as the second command.
- For example, if the user enters *ls -l*, ls will be the first argument of the system call whereas the entire string will be the second argument.

Part 2

- Implement a history feature
- Execute the most recently executed command again. Make sure the command is first displayed and then executed
- If no command has been most recently executed, the terminal should display: "No commands in history"



Part 3

```
osh>echo "Hello" > Hello
osh>cat Hello
"Hello"
osh>
```

Output Redirection

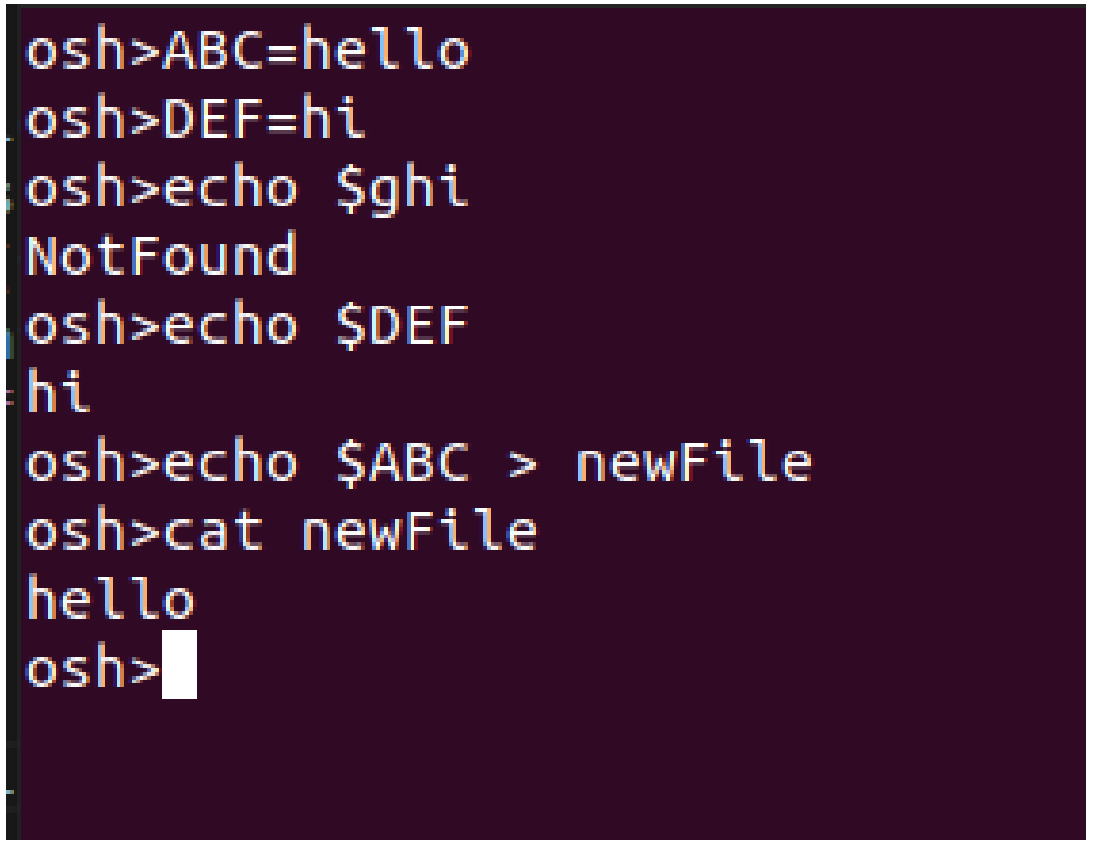
Your program should now support '>' operator, called the output redirection operator.

It redirects the output of a program to a file.

Hint: You need to use file descriptors along with the dup2 system call

Part 4

- In this part, you must program your shell to support environment variables
- Once, environment variables are declared and assigned values, our program should be able to access them, print their values using the echo command or redirect their values to a file



Part 5

```
osh>cat simple-shell.c | head -5
/**
 * Simple shell interface program.
 *
 * Operating System Concepts - Tenth Edition
 * Copyright John Wiley & Sons - 2018
osh>
```

Inter process Communication

Two processes will communicate via the UNIX pipe() function, where the output of one process will be the input of another process.

Create a child process to execute one command. This child process creates another child process to execute the second command. The two processes communicate via pipe.

Questions?