# Walking a File Tree in Popular Languages

## Table of Contents

## Python

### Using `os.walk`

```python
import os

for root, dirs, files in os.walk("/path/to/directory"):
    print(f"Directory: {root}")
    print(f"Subdirectories: {dirs}")
    print(f"Files: {files}")
```

### Using `pathlib`

```python
from pathlib import Path

for path in Path("/path/to/directory").rglob("*"):
    print(path)
```

## C++

```cpp
#include <iostream>
#include <filesystem>
namespace fs = std::filesystem;

int main() {
    for (const auto& entry : fs::recursive_directory_iterator("/path/to/directory")
        ) {
        std::cout << entry.path() << std::endl;
    }
    return 0;
}
```

## Zsh

### Recursive Globbing with `**`

```zsh
for file in /path/to/directory/**/*; do
  echo $file
done
```

### Using `find`

```zsh
for file in $(find /path/to/directory -type f); do
  echo $file
done
```

### Using Built-in Functions

```
walk_tree() {
  local dir=$1
  for file in $dir/**/*; do
    [[ -f $file ]] && echo "File: $file"
    [[ -d $file ]] && echo "Directory: $file"
  done
}

walk_tree /path/to/directory
```

## Bash

```
find /path/to/directory -type f -print
```

## JavaScript (Node.js)

```
const fs = require('fs');
const path = require('path');

function walk(dir) {
    fs.readdirSync(dir).forEach(file => {
        const fullPath = path.join(dir, file);
        if (fs.statSync(fullPath).isDirectory()) {
            walk(fullPath);
        } else {
            console.log(fullPath);
        }
    });
}

walk('/path/to/directory');
```

## Rust

```
use walkdir::WalkDir;

fn main() {
    for entry in WalkDir::new("/path/to/directory") {
        let entry = entry.unwrap();
        println!("{}", entry.path().display());
    }
}
```

## Zig

```
const std = @import("std");

pub fn main() !void {
    const cwd = try std.fs.cwd();
    var it = try cwd.walk("/path/to/directory", .{});
    while (try it.next()) |entry| {
        if (entry.kind == .File) {
            std.debug.print("File: {s}\n", .{entry.path});
        } else if (entry.kind == .Directory) {
            std.debug.print("Directory: {s}\n", .{entry.path});
        }
    }
}
```

## Go

```
package main

import (
        "fmt"
        "os"
        "path/filepath"
)

func main() {
        root := "/path/to/directory"
        err := filepath.Walk(root, func(path string, info os.FileInfo, err error)
            error {
                if err != nil {
                        return err
                }
                fmt.Println(path)
                return nil
        })
        if err != nil {
                fmt.Println("Error:", err)
        }
}
```

## C#

### Using Directory.GetFiles

```
using System;
using System.IO;

class Program {
    static void Main() {
        foreach (string file in Directory.GetFiles("/path/to/directory", "*",
            SearchOption.AllDirectories)) {
            Console.WriteLine(file);
        }
    }
}
```

## Java

```
import java.nio.file.*;
import java.io.IOException;

public class FileTreeWalk {
    public static void main(String[] args) throws IOException {
        Path path = Paths.get("/path/to/directory");
        Files.walk(path).forEach(System.out::println);
    }
}
```

## Ruby

### Using `Dir.glob`

```ruby
Dir.glob("/path/to/directory/**/*").each do |file|
  puts file
end
```

### Using `Find`

```ruby
require 'find'

Find.find('/path/to/directory') do |path|
  puts path
end
```

## PHP

```php
<?php
$iterator = new RecursiveIteratorIterator(new RecursiveDirectoryIterator('/path/to/
    directory'));
foreach ($iterator as $file) {
    echo $file . PHP_EOL;
}
?>
```