12-fortuner

目标：实现 fortune 命令

实现

- 解析数字

Rust

```rust
fn parse_u64(val: &str) -> MyResult<u64> {
    val.parse()
        .map_err(|_| format!("\"{}\" not a valid integer", val).into())
}
```

- 查找文件

Rust

```rust
fn find_files(paths: &[String]) -> MyResult<Vec<PathBuf>> {
    let dat = OsStr::new("dat");
    let mut files = vec![];

    for path in paths {
        match fs::metadata(path) {
            Err(e) => return Err(format!("{}: {}", path, e).into()),
            Ok(_) => files.extend(
                WalkDir::new(path)
                    .into_iter()
                    .filter_map(Result::ok)
                    .filter(|e| {
                        e.file_type().is_file()
                            && e.path().extension() != Some(dat)
                    })
                    .map(|e| e.path().into()),
            ),
        }
    }

    files.sort();
    files.dedup();
    Ok(files)
}
```

- Create an `OsStr` value for the string dat.
- Create a mutable vector for the results.
- If `fs::metadata` fails, return a useful error message.
- Use `Vec::extend` to add the results from `WalkDir` to the results.（将结果插入到files里）
- Use `walkdir::WalkDir` to find all the entries from the starting path.
- `filter_map` will ignore any errors for unreadable files or directories, which is the behavior of the original program.
- Take only regular files that do not have the .dat extension.
- The `walkdir::DirEntry::path` function returns a Path, so convert it into a `PathBuf`.
- Use `Vec::sort` to sort the entries in place.
- Use `Vec::dedup` to remove consecutive repeated values.（去重）

- 读取fortunes

Rust

```rust
fn read_fortunes(paths: &[PathBuf]) -> MyResult<Vec<Fortune>> {
    let mut fortunes = vec![];
    let mut buffer = vec![];

    for path in paths {
        let basename =
            path.file_name().unwrap().to_string_lossy().into_owned();
        let file = File::open(path).map_err(|e| {
            format!("{}: {}", path.to_string_lossy().into_owned(), e)
        })?;

        for line in BufReader::new(file).lines().filter_map(Result::ok) {
            if line == "%" {
                if !buffer.is_empty() {
                    fortunes.push(Fortune {
                        source: basename.clone(),
                        text: buffer.join("\n"),
                    });
                    buffer.clear();
                }
            } else {
                buffer.push(line.to_string());
            }
        }
    }

    Ok(fortunes)
}
```

- Convert `Path::file_name` from `OsStr` to `String`, using the lossy version in case this is not valid UTF-8. The result is a clone-on-write smart pointer, so use `Cow::into_owned` to clone the data if it is not already owned.

- 挑选fortune

Rust

```rust
fn pick_fortune(fortunes: &[Fortune], seed: Option<u64>) -> Option<String> {
    if let Some(val) = seed {
        let mut rng = StdRng::seed_from_u64(val);
        fortunes.choose(&mut rng).map(|f| f.text.to_string())
    } else {
        let mut rng = rand::thread_rng();
        fortunes.choose(&mut rng).map(|f| f.text.to_string())
    }
}
```