dirtreex – Examples Gallery

This document walks through the full feature surface of the `dirtreex` package. Each numbered entry shows (a) the raw source code exactly as you would write it, and (b) the rendered result produced by that code. The examples are self-contained: copy any snippet into a preamble with `\usepackage{dirtreex}` and it will compile.

Contents

1 Basics

1.1 Default tree

```
\begin{dirtreex}
  \dir{project}{root directory}{
    \dir{src}{source code}{
      \file{main.py}{entry point}
      \file{util.py}{helpers}
    }
    \file{README.md}{documentation}
  }
\end{dirtreex}
```

```
project/ ..... root directory
├── src/ ..... source code
│   ├── main.py ..... entry point
│   └── util.py ..... helpers
└── README.md ..... documentation
```

1.2 Empty comments and empty subtrees

```
\begin{dirtreex}
  \dir{root}{}{
    \file{a.txt}{with comment}
    \file{b.txt}{}
    \dir{empty-dir}{}{}
    \dir{has-comment}{no children}{}
  }
\end{dirtreex}
```

```
root/
├── a.txt ..... with comment
├── b.txt
├── empty-dir/
└── has-comment/ ..... no children
```

1.3 Single-node tree

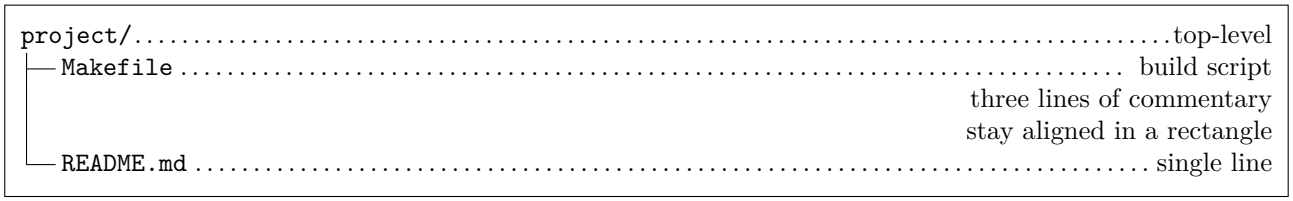
```
\begin{dirtreex}
  \file{lonely.txt}{a single node is legal}
\end{dirtreex}
```

```
lonely.txt ..... a single node is legal
```

1.4 Multi-line comment

```
\begin{dirtreex}
  \dir{project}{top-level}{
    \file{Makefile}{
```

```
      build script\\
      three lines of commentary\\
      stay aligned in a rectangle%
    }
    \file{README.md}{single line}
  }
\end{dirtreex}
```



2 Archive nodes

`\archive` is shape-identical to `\dir` (takes a children block, accepts the same per-entry override keys) but renders its name *without* the trailing `/`. Use it for archive-like containers (`.tar.gz`, `.zip`, `.jar`, …) whose contents you still want to display nested in the tree. Internally the three entry commands are distinguished by the `t` slot — `\dir` stores 1, `\file` stores 0, `\archive` stores 2 — which the public hook `\DirtreexFormatName` reads to decide whether to append `/`.

2.1 Default archive at the top level

```
\begin{dirtreex}
  \archive{release-2026.tar.gz}{shipped tarball}{
    \file{manifest.json}{contents listing}
    \file{LICENSE}{}
    \dir{bin}{executables}{
      \file{dirtreex.sty}{}
    }
  }
\end{dirtreex}
```

```
release-2026.tar.gz ..... shipped tarball
├── manifest.json ..... contents listing
├── LICENSE
└── bin/ ..... executables
    └── dirtreex.sty
```

Compare the root entry `release-2026.tar.gz` (no trailing `/`) with the nested `bin/` directory underneath it.

2.2 Archive nested inside a directory

Archives mix freely with directories at any depth. The depth tracker treats `\archive` as a container just like `\dir`, so the elbow geometry and active-trunk bookkeeping carry over unchanged.

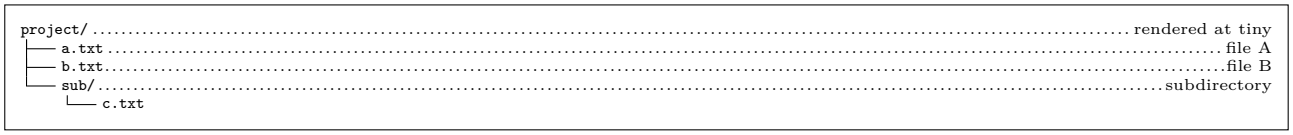
```
\begin{dirtreex}
  \dir{project}{root}{
    \dir{dist}{distribution outputs}{
      \archive{project-1.0.zip}{user download}{
        \file{README.md}{}
        \file{install.sh}{}
      }
      \archive{project-1.0-src.tar.gz}{source bundle}{
        \dir{src}{}{
          \file{main.c}{}
        }
      }
    }
    \file{Makefile}{}
  }
\end{dirtreex}
```

```
project/ ..... root
├── dist/ ..... distribution outputs
│   ├── project-1.0.zip ..... user download
│   │   ├── README.md
│   │   └── install.sh
│   └── project-1.0-src.tar.gz ..... source bundle
│       └── src/
│           └── main.c
└── Makefile
```

3 Font size

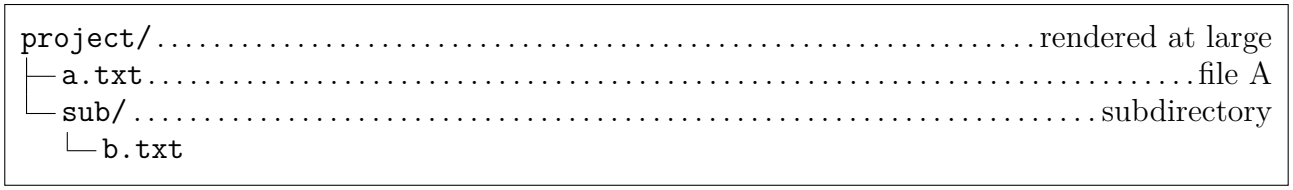
3.1 Tiny font

```
\begin{dirtreex}[fontsize=\tiny]
  \dir{project}{rendered at tiny}{
    \file{a.txt}{file A}
    \file{b.txt}{file B}
    \dir{sub}{subdirectory}{\file{c.txt}{}}
  }
\end{dirtreex}
```



3.2 Large font

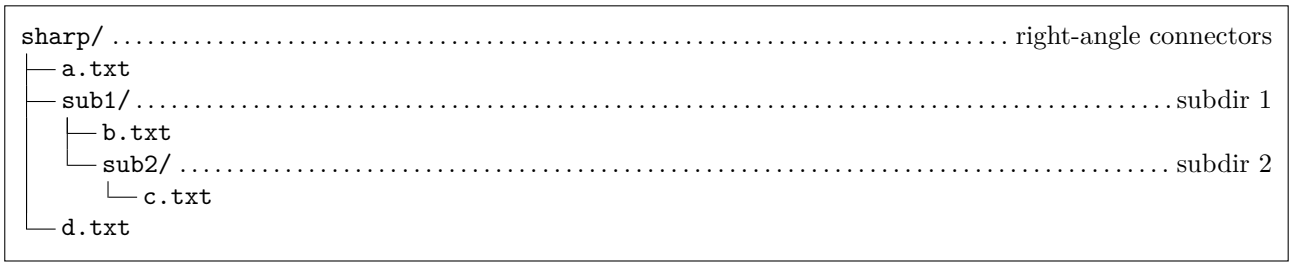
```
\begin{dirtreex}[fontsize=\large]
  \dir{project}{rendered at large}{
    \file{a.txt}{file A}
    \dir{sub}{subdirectory}{\file{b.txt}{}}
  }
\end{dirtreex}
```



4 Elbow shape

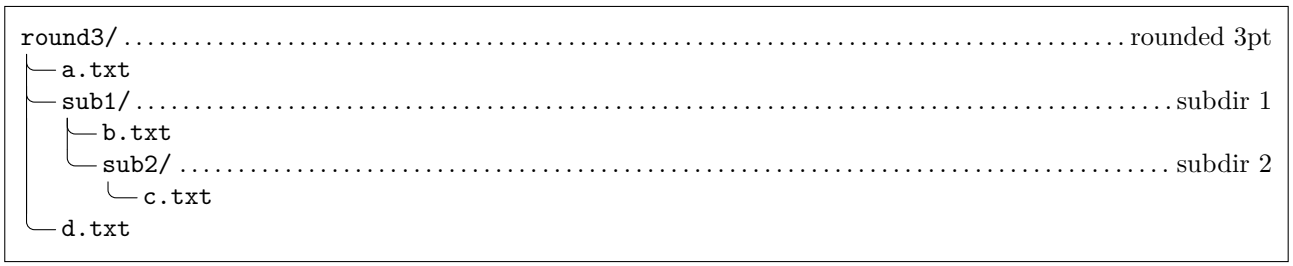
4.1 Sharp elbows (default)

```
\begin{dirtreex}[elbow radius=0pt]
  \dir{sharp}{right-angle connectors}{
    \file{a.txt}{}
    \dir{sub1}{subdir 1}{
      \file{b.txt}{}
      \dir{sub2}{subdir 2}{\file{c.txt}{}}
    }
    \file{d.txt}{}
  }
\end{dirtreex}
```



4.2 Rounded elbows, radius 3pt

```
\begin{dirtreex}[elbow radius=3pt]
  \dir{round3}{rounded 3pt}{
    \file{a.txt}{}
    \dir{sub1}{subdir 1}{
      \file{b.txt}{}
      \dir{sub2}{subdir 2}{\file{c.txt}{}}
    }
    \file{d.txt}{}
  }
\end{dirtreex}
```



4.3 Rounded elbows, radius 6pt

```
\begin{dirtreex}[elbow radius=6pt]
  \dir{round6}{rounded 6pt}{
    \file{a.txt}{}
    \dir{sub1}{subdir 1}{
      \file{b.txt}{}
      \dir{sub2}{subdir 2}{\file{c.txt}{}}
    }
    \file{d.txt}{}
```

```
  }
\end{dirtreex}
```

round6/ ..... rounded 6pt
- a.txt
- sub1/ ..... subdir 1
  - b.txt
  - sub2/ ..... subdir 2
    - c.txt
- d.txt

4.4 Extreme radius (clamped to `0.5\baselineskip`)

The package clamps the requested elbow radius against half the baseline skip (and against the line width), so an oversized value silently degrades to the largest legible arc rather than overrunning the column.

```
\begin{dirtreex}[elbow radius=10pt]
  \dir{round10}{requested 10pt -- clamped}{
    \file{a.txt}{}
    \dir{sub1}{subdir 1}{
      \file{b.txt}{}
      \dir{sub2}{subdir 2}{\file{c.txt}{}}
    }
    \file{d.txt}{}
  }
\end{dirtreex}
```

round10/ ..... requested 10pt – clamped
- a.txt
- sub1/ ..... subdir 1
  - b.txt
  - sub2/ ..... subdir 2
    - c.txt
- d.txt

5 Frame (the box family)

5.1 No frame (box=false)

```
\begin{dirtreex}[box=false]
  \dir{bare}{no border, no padding}{
    \file{a.txt}{}
    \dir{sub}{}{\file{b.txt}{}}
  }
\end{dirtreex}
```

bare/ ..... no border, no padding
├── a.txt
└── sub/
    └── b.txt

5.2 Rounded corners (uniform)

```
\begin{dirtreex}[
  box={true, corners=6pt, border color=black,
       background color=white, margin=6pt}]
  \dir{rounded-frame}{corners=6pt}{
    \file{a.txt}{}
    \dir{sub}{}{\file{b.txt}{}}
  }
\end{dirtreex}
```

rounded-frame/ ..... corners=6pt
├── a.txt
└── sub/
    └── b.txt

5.3 Coloured border and background

```
\begin{dirtreex}[
  box={true, corners=3pt, border color=purple,
       border width=1pt, background color=purple!5}]
  \dir{colored}{purple border, lavender fill}{
    \file{a.txt}{}
    \file{b.txt}{}
  }
\end{dirtreex}
```

colored/ ..... purple border, lavender fill
├── a.txt
└── b.txt

5.4 HTML hex colour + thick border

```
\definecolor{brand}{HTML}{2C7BE5}
\begin{dirtreex}[
```

```
  box={true, corners=4pt, border color=brand,
       border width=1.5pt,
       background color=brand!8}]
  \dir{hex-color}{\#2C7BE5 accent}{
    \file{main.ts}{}
    \file{style.css}{}
  }
\end{dirtreex}
```

hex-color/ ..... #2C7BE5 accent
├── main.ts
└── style.css

5.5 Four independent corner radii

```
\begin{dirtreex}[
  box={true, corners={8pt, 0pt, 8pt, 0pt},
       border color=teal, background color=teal!5,
       margin=6pt}]
  \dir{corners}{TL, TR, BR, BL = 8, 0, 8, 0}{
    \file{a.txt}{}
    \file{b.txt}{}
  }
\end{dirtreex}
```

corners/ ..... TL, TR, BR, BL = 8, 0, 8, 0
├── a.txt
└── b.txt

5.6 Uniform margin (single value)

```
\begin{dirtreex}[
  box={margin=18pt, border color=black,
       background color=gray!5}]
  \dir{padded}{18pt margin on every side}{
    \file{a.txt}{}
    \file{b.txt}{}
  }
\end{dirtreex}
```

padded/ ..... 18pt margin on every side
├── a.txt
└── b.txt

5.7 Asymmetric margin (four values, T/R/B/L)

```
\begin{dirtreex}[
  box={margin={2pt, 30pt, 16pt, 4pt},
       border color=orange, background color=orange!8}]
```

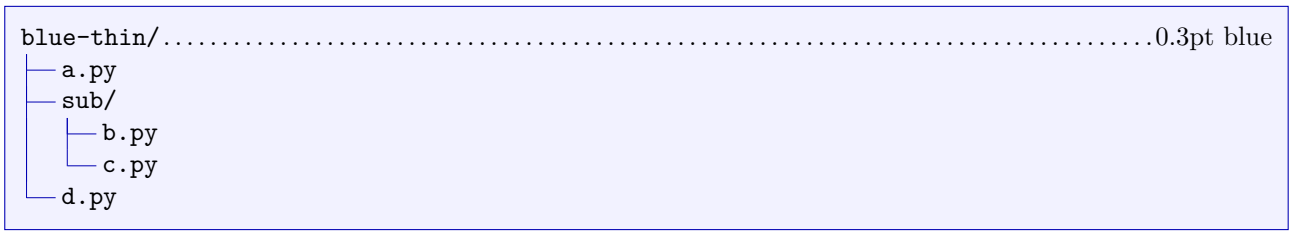
```
  \dir{asymmetric}{T=2, R=30, B=16, L=4}{
    \file{a}{}
    \file{b}{}
  }
\end{dirtreex}
```

asymmetric/ ..... T=2, R=30, B=16, L=4
├── a
└── b

6 Global line style

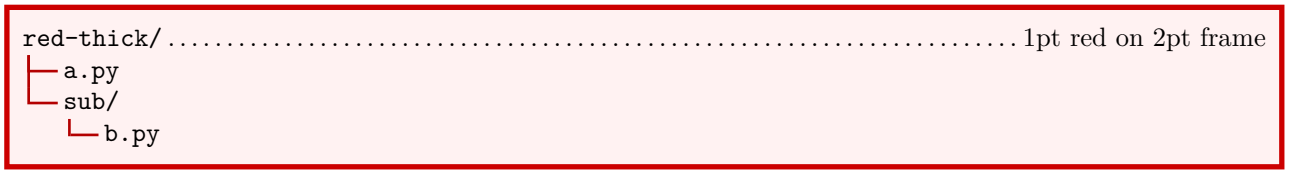
6.1 Thin blue lines

```
\begin{dirtreex}[
  line color=blue!70!black, line width=0.3pt,
  box={true, border color=blue!70!black,
       background color=blue!5, margin=6pt}]
  \dir{blue-thin}{0.3pt blue}{
    \file{a.py}{}
    \dir{sub}{}{
      \file{b.py}{}
      \file{c.py}{}
    }
    \file{d.py}{}
  }
\end{dirtreex}
```



6.2 Thick red lines

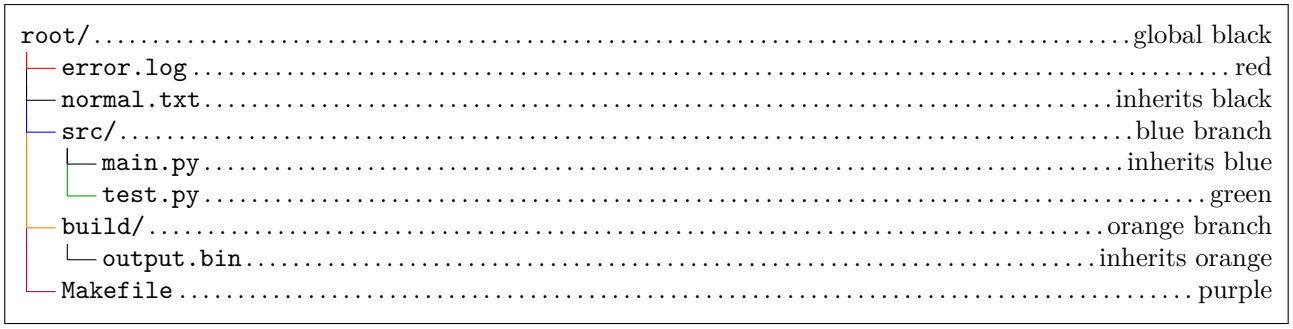
```
\begin{dirtreex}[
  line color=red!70!black, line width=1pt,
  box={true, border color=red!80!black,
       border width=2pt, background color=red!5,
       margin=6pt}]
  \dir{red-thick}{1pt red on 2pt frame}{
    \file{a.py}{}
    \dir{sub}{}{\file{b.py}{}}
  }
\end{dirtreex}
```



7 Per-entry overrides

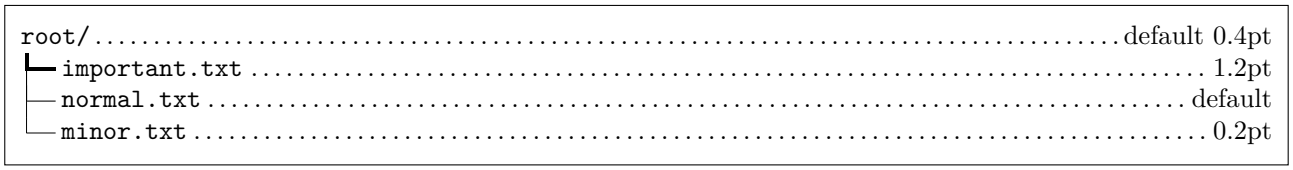
7.1 Line colour override

```
\begin{dirtreex}
  \dir{root}{global black}{
    \file[line color=red]{error.log}{red}
    \file{normal.txt}{inherits black}
    \dir[line color=blue]{src}{blue branch}{
      \file{main.py}{inherits blue}
      \file[line color=green!60!black]{test.py}{green}
    }
    \dir[line color=orange]{build}{orange branch}{
      \file{output.bin}{inherits orange}
    }
    \file[line color=purple]{Makefile}{purple}
  }
\end{dirtreex}
```



7.2 Line width override

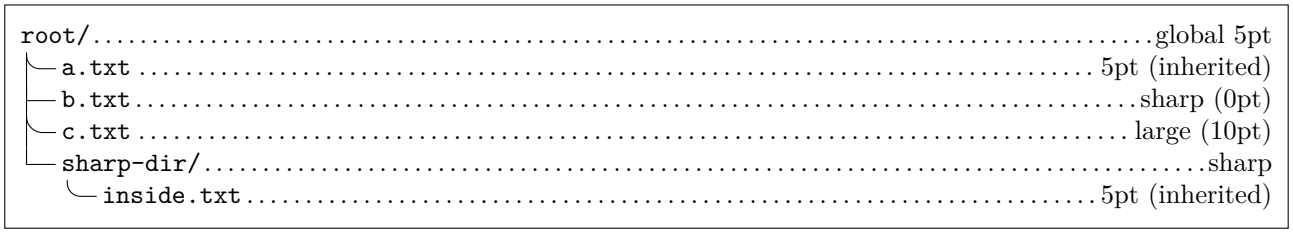
```
\begin{dirtreex}
  \dir{root}{default 0.4pt}{
    \file[line width=1.2pt]{important.txt}{1.2pt}
    \file{normal.txt}{default}
    \file[line width=0.2pt]{minor.txt}{0.2pt}
  }
\end{dirtreex}
```



7.3 Elbow radius override

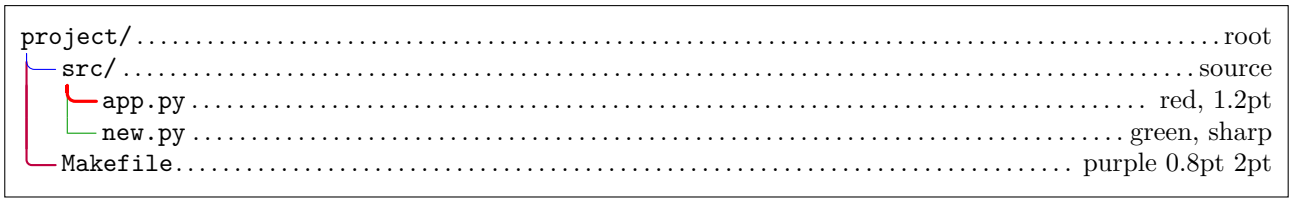
```
\begin{dirtreex}[elbow radius=5pt]
  \dir{root}{global 5pt}{
    \file{a.txt}{5pt (inherited)}
    \file[elbow radius=0pt]{b.txt}{sharp (0pt)}
    \file[elbow radius=10pt]{c.txt}{large (10pt)}
    \dir[elbow radius=0pt]{sharp-dir}{sharp}{
      \file{inside.txt}{5pt (inherited)}
    }
```

```
  }
\end{dirtreex}
```



7.4 Combined overrides (colour + width + elbow)

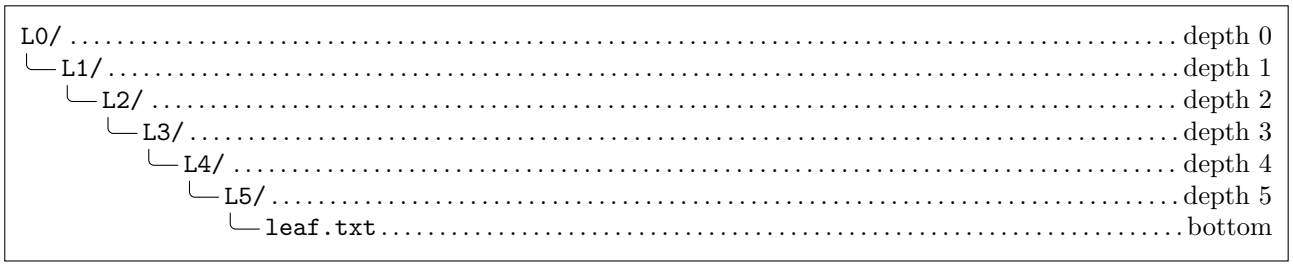
```
\begin{dirtreex}[elbow radius=3pt]
  \dir{project}{root}{
    \dir[line color=blue]{src}{source}{
      \file[line color=red, line width=1.2pt]{app.py}{
        red, 1.2pt%
      }
      \file[line color=green!60!black,
            elbow radius=0pt]{new.py}{green, sharp}
    }
    \file[line color=purple, line width=0.8pt,
          elbow radius=2pt]{Makefile}{
      purple 0.8pt 2pt%
    }
  }
\end{dirtreex}
```



8 Deep nesting

8.1 Six-level tree

```
\begin{dirtreex}[elbow radius=2pt]
  \dir{L0}{depth 0}{
    \dir{L1}{depth 1}{
      \dir{L2}{depth 2}{
        \dir{L3}{depth 3}{
          \dir{L4}{depth 4}{
            \dir{L5}{depth 5}{
              \file{leaf.txt}{bottom}
            }
          }
        }
      }
    }
  }
\end{dirtreex}
```



8.2 Eight-level tree with a shallow sibling

Mixing a deep branch and a shallow sibling shows that the per-depth column stride is uniform: every level contributes one connector slot, and the shallow sibling sits at the right offset without any manual spacing.

```
\begin{dirtreex}[elbow radius=2pt]
  \dir{L0}{depth 0}{
    \dir{L1}{}{
      \dir{L2}{}{
        \dir{L3}{}{
          \dir{L4}{}{
            \dir{L5}{}{
              \dir{L6}{}{
                \dir{L7}{depth 7}{
                  \file{leaf.txt}{deepest}
                }
              }
            }
          }
        }
      }
    }
    \file{shallow.txt}{level-1 sibling}
  }
\end{dirtreex}
```

```
L0/ ..... depth 0
└── L1/
    └── L2/
        └── L3/
            └── L4/
                └── L5/
                    └── L6/
                        └── L7/ ..... depth 7
                            └── leaf.txt ..... deepest
└── shallow.txt ..... level-1 sibling
```

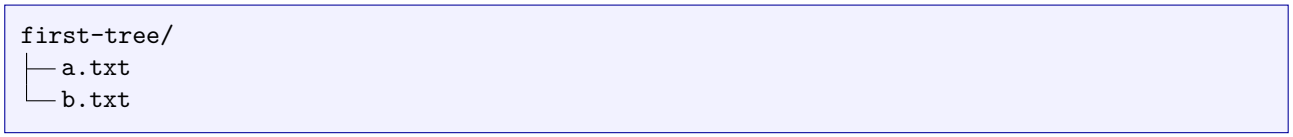
9 Multiple trees

9.1 Two trees in sequence

```
\begin{dirtreex}[box={border color=blue!60!black,
                      background color=blue!5}]
  \dir{first-tree}{}{
    \file{a.txt}{}
    \file{b.txt}{}
  }
\end{dirtreex}

\bigskip

\begin{dirtreex}[box={border color=red!60!black,
                      background color=red!5}]
  \dir{second-tree}{}{
    \file{x.py}{}
    \file{y.py}{}
  }
\end{dirtreex}
```



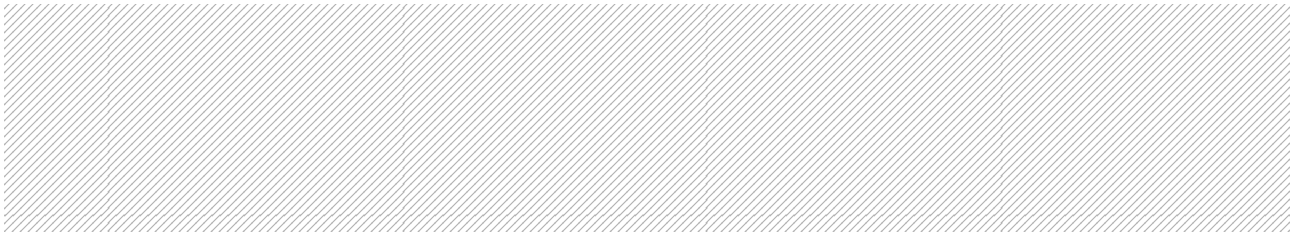
10 Page-break behaviour

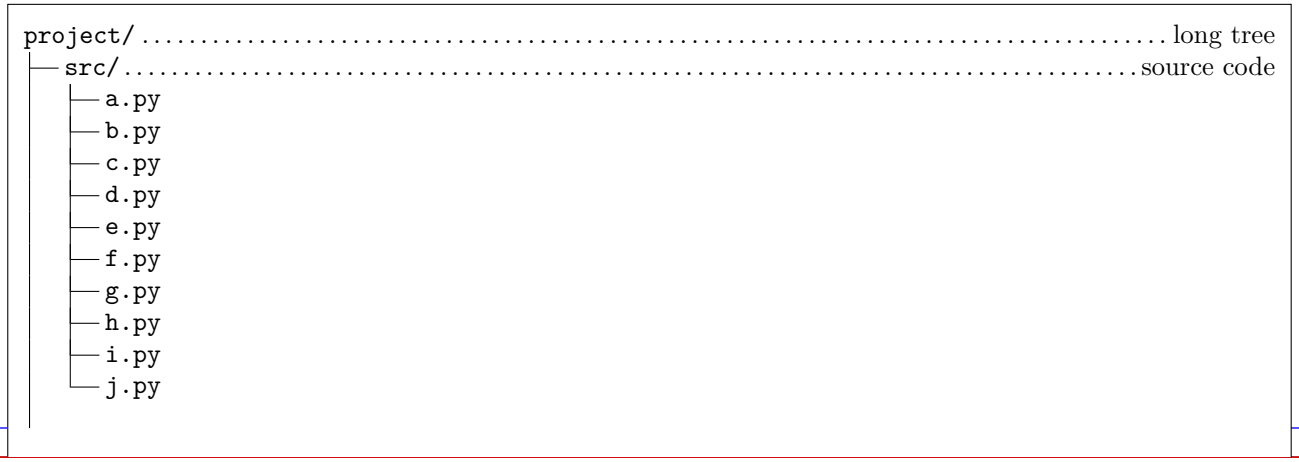
The examples below all need enough content to overflow onto a second page. They are placed after intentional vertical filler so the tree must split at least once; in real documents the break is driven by whatever natural layout pushes the tree close to a page boundary.

The black line outlines the content area, while the red and blue lines indicate where the box and directory tree should break, respectively.

10.1 Default cross-page rendering

```
\begin{dirtreex}[
  box={true, border color=black,
       background color=white, margin=6pt}]
  \dir{project}{long tree}{
    \dir{src}{source code}{
      \file{a.py}{}\file{b.py}{}\file{c.py}{}
      \file{d.py}{}\file{e.py}{}\file{f.py}{}
      \file{g.py}{}\file{h.py}{}\file{i.py}{}
      \file{j.py}{}
    }
    \dir{tests}{test suite}{
      \file{test-a.py}{}\file{test-b.py}{}
      \file{test-c.py}{}\file{test-d.py}{}
      \file{test-e.py}{}
    }
    \dir{docs}{documentation}{
      \file{intro.md}{}\file{api.md}{}
      \file{tutorial.md}{}\file{changelog.md}{}
    }
    \file{Makefile}{}
    \file{setup.py}{}
    \file{README.md}{}
    \file{LICENSE}{}
  }
\end{dirtreex}
```

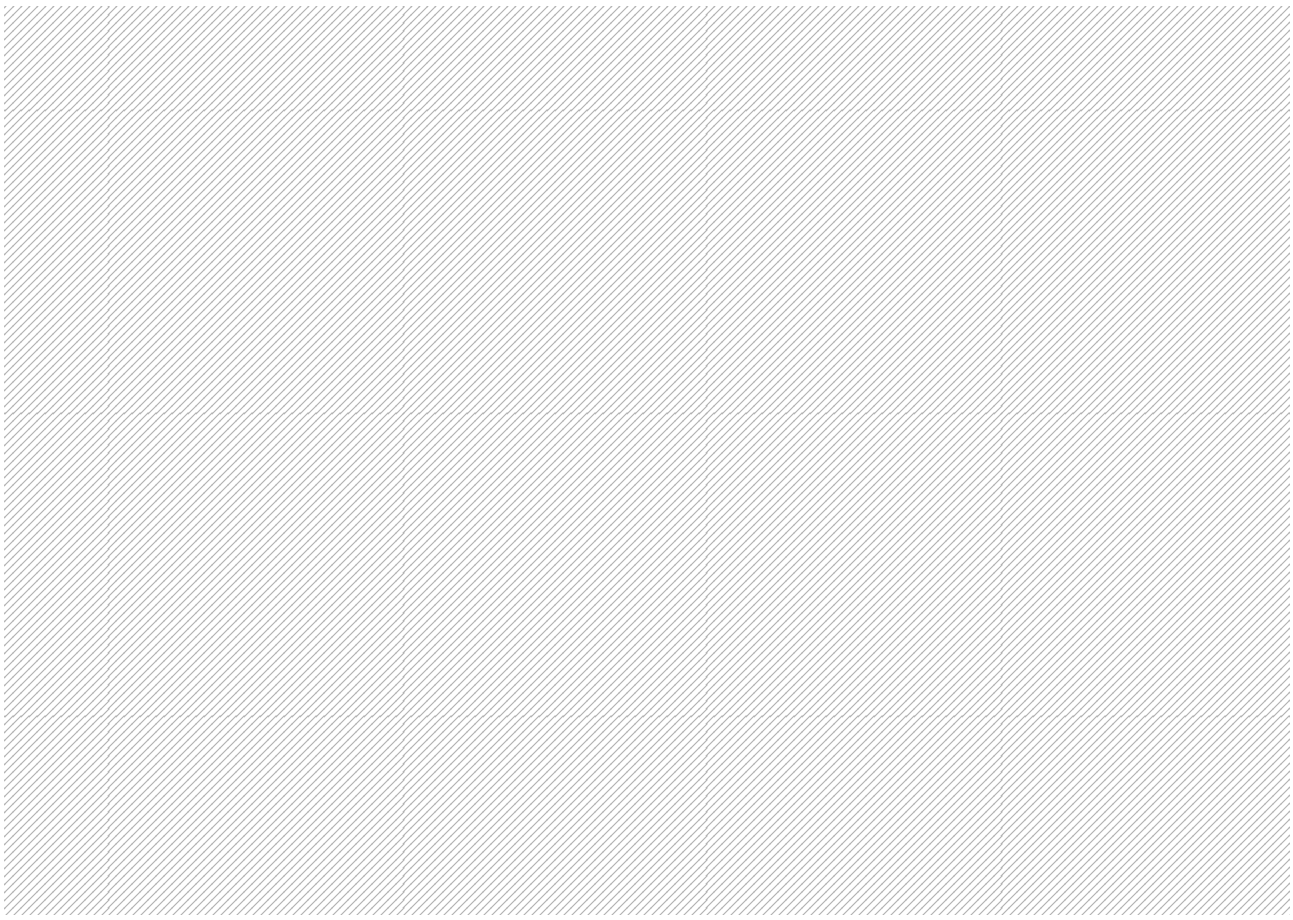




```
├── tests/ ..... test suite
│   ├── test-a.py
│   ├── test-b.py
│   ├── test-c.py
│   ├── test-d.py
│   └── test-e.py
├── docs/ ..... documentation
│   ├── intro.md
│   ├── api.md
│   ├── tutorial.md
│   └── changelog.md
├── Makefile
├── setup.py
├── README.md
└── LICENSE
```

10.2 Cross-page with rounded elbows and per-entry colours

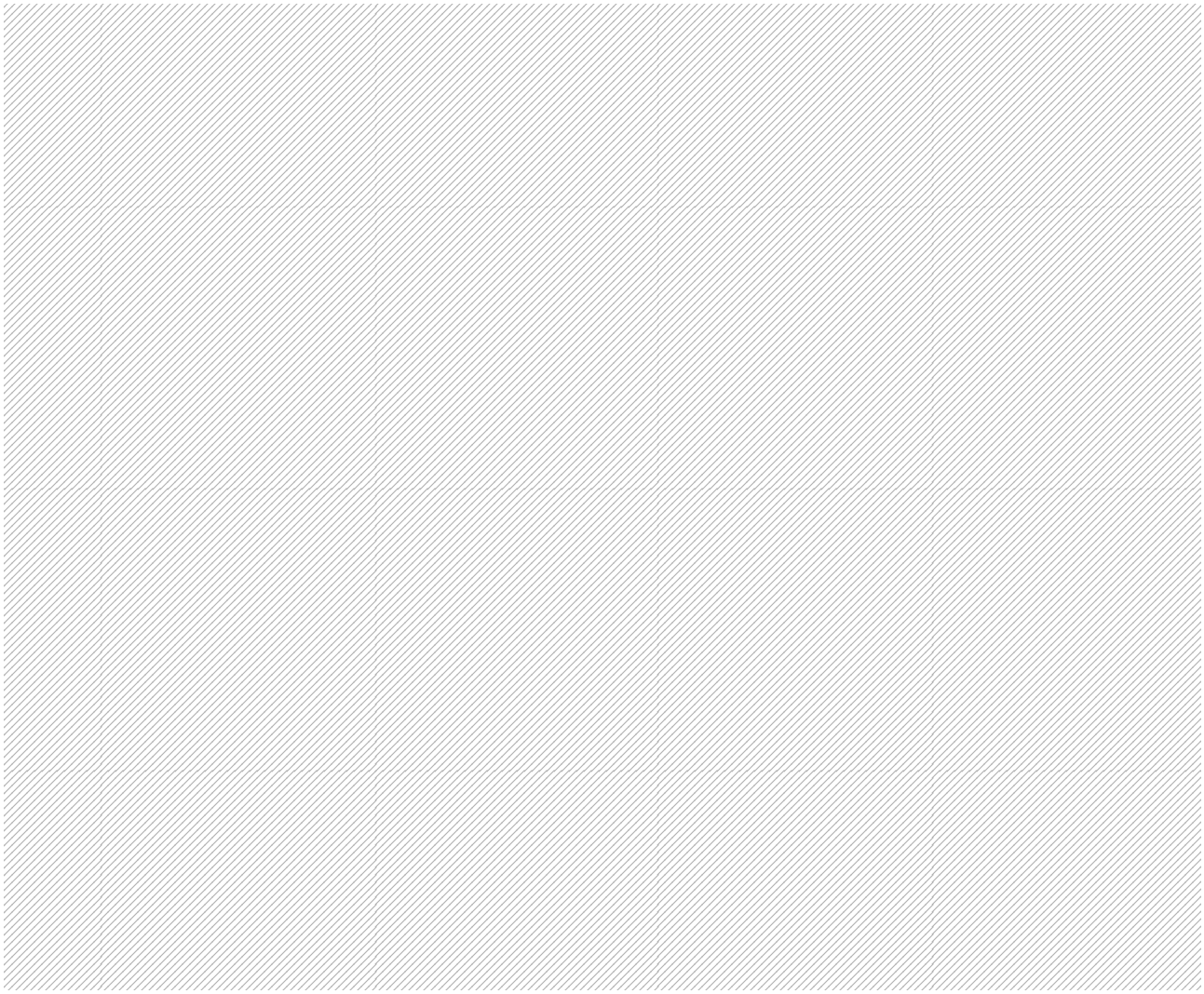
```
\begin{dirtreex}[
  elbow radius=3pt,
  box={true, corners=4pt, border color=black,
       background color=white, margin=6pt}]
  \dir{project}{colour across a page break}{
    \dir[line color=red]{red-branch}{red}{
      \file{r1.py}{}\file{r2.py}{}\file{r3.py}{}
      \file{r4.py}{}\file{r5.py}{}\file{r6.py}{}
      \file{r7.py}{}\file{r8.py}{}\file{r9.py}{}
      \file{r10.py}{}
    }
    \dir[line color=blue]{blue-branch}{blue}{
      \file{b1.py}{}\file{b2.py}{}\file{b3.py}{}
      \file{b4.py}{}\file{b5.py}{}
    }
    \dir[line color=green!60!black]{green-branch}{green}{
      \file{g1.py}{}\file{g2.py}{}\file{g3.py}{}
    }
  }
\end{dirtreex}
```



```
    ├── r6.py
    ├── r7.py
    ├── r8.py
    ├── r9.py
    └── r10.py
├── blue-branch/ ..... blue
    ├── b1.py
    ├── b2.py
    ├── b3.py
    ├── b4.py
    └── b5.py
└── green-branch/ ..... green
    ├── g1.py
    ├── g2.py
    └── g3.py
```

10.3 Break offsets (`box break at` / `tree break at`)

```
\begin{dirtreex}[
  box={true, corners=4pt, border color=blue!60!black,
       background color=blue!5, margin=6pt},
  pagebreak={true, box break at=1em,
             tree break at=2em}]
  \dir{offsets}{box pulls back 4pt, tree pulls back 2pt}{
    \dir{src}{}{
      \file{a.py}{}\file{b.py}{}\file{c.py}{}
      \file{d.py}{}\file{e.py}{}\file{f.py}{}
      \file{g.py}{}\file{h.py}{}
    }
    \dir{docs}{}{
      \file{x.md}{}\file{y.md}{}\file{z.md}{}
    }
  }
\end{dirtreex}
```

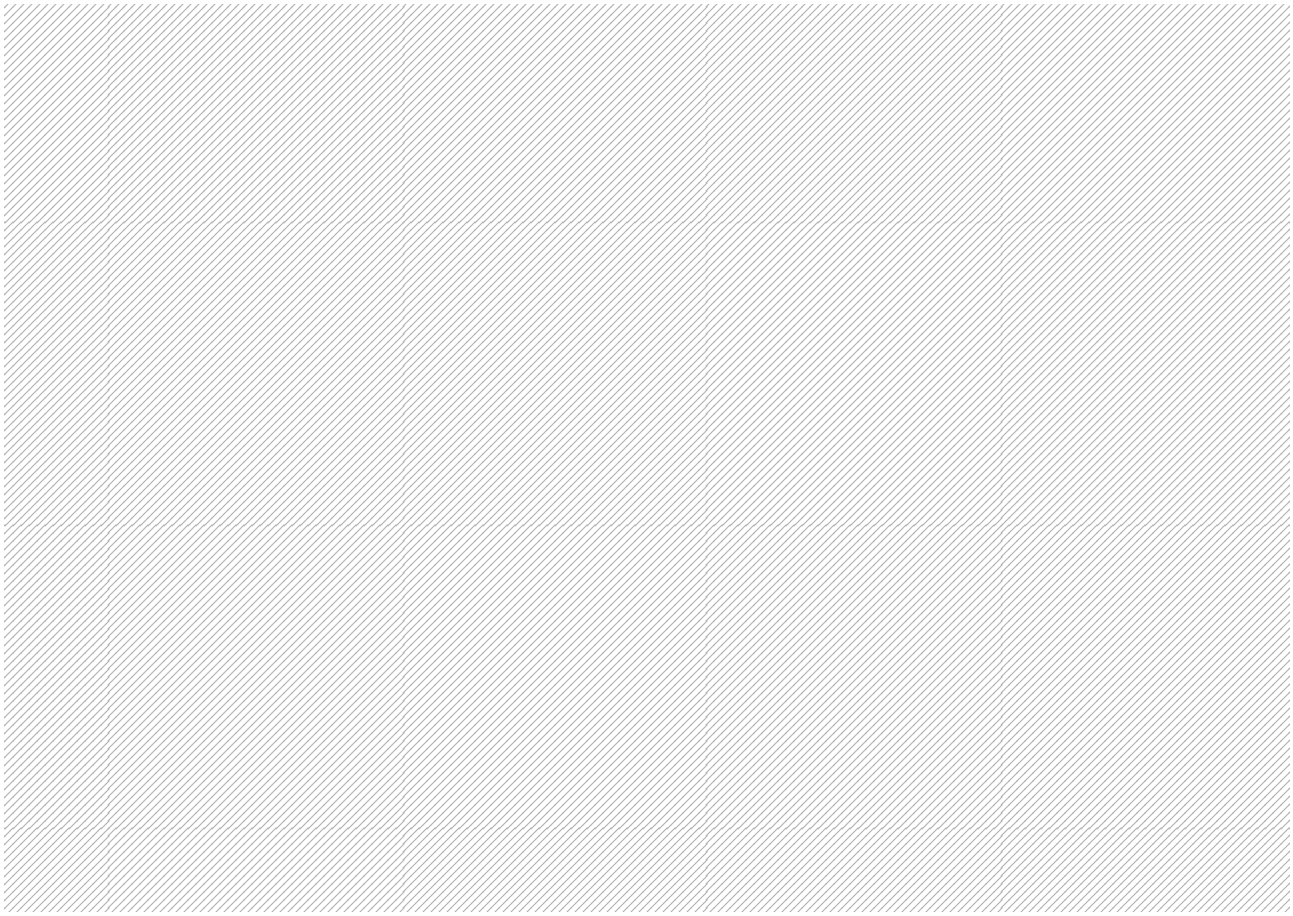


offsets/ ..... box pulls back 1em, tree pulls back 2em
- src/
  - a.py
  - b.py
  - c.py

```
    ├── d.py
    ├── e.py
    ├── f.py
    ├── g.py
    └── h.py
└── docs/
    ├── x.md
    ├── y.md
    └── z.md
```

10.4 Asymmetric break offsets (two values)

```
\begin{dirtreex}[
  box={true, corners=4pt, border color=purple!60!black,
       background color=purple!5, margin=6pt},
  pagebreak={true, box break at={2em, 1em},
             tree break at={4em, 3em}}]
  \dir{asym}{first-piece-bottom=2em and 4em, next-piece-top=1em and 3em}{
    \dir{src}{}{
      \file{a.py}{}\file{b.py}{}\file{c.py}{}
      \file{d.py}{}\file{e.py}{}\file{f.py}{}
      \file{g.py}{}\file{h.py}{}
    }
    \dir{lib}{}{
      \file{x.py}{}\file{y.py}{}
    }
  }
\end{dirtreex}
```



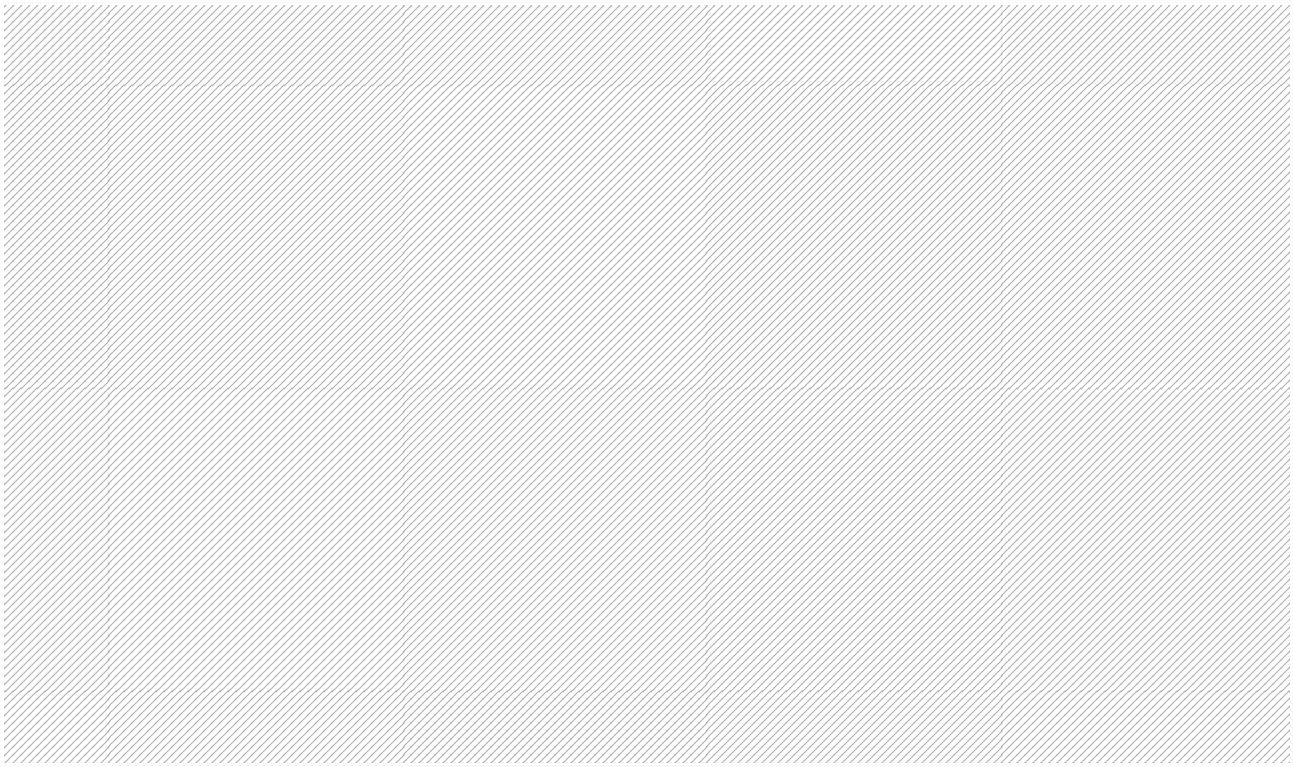
asym/ ..... first-piece-bottom=2em and 4em, next-piece-top=1em and 3em
- src/
  - a.py
  - b.py
  - c.py
  - d.py
  - e.py

```
    ├── f.py
    ├── g.py
    └── h.py
└── lib/
    ├── x.py
    └── y.py
```

10.5 Bare tree (`box=false`) across a page break

With `box=false` there is no frame to draw, but the tree's own extension columns still need to stay coherent across the break: every active `|` continues to the page bottom, then resumes at the top of the continuation page so descendants under a parent on the previous page remain visually attached. `box break at` is ignored when the frame is off (there is no torn edge to pull back).

```
\begin{dirtreex}[box=false]
  \dir{bare-cross}{no frame, tree still breaks}{
    \dir{src}{}{
      \file{a.py}{}\file{b.py}{}\file{c.py}{}
      \file{d.py}{}\file{e.py}{}\file{f.py}{}
      \file{g.py}{}\file{h.py}{}\file{i.py}{}
    }
    \dir{tests}{}{
      \file{t1.py}{}\file{t2.py}{}\file{t3.py}{}
      \file{t4.py}{}
    }
    \file{README.md}{}\file{LICENSE}{}
  }
\end{dirtreex}
```



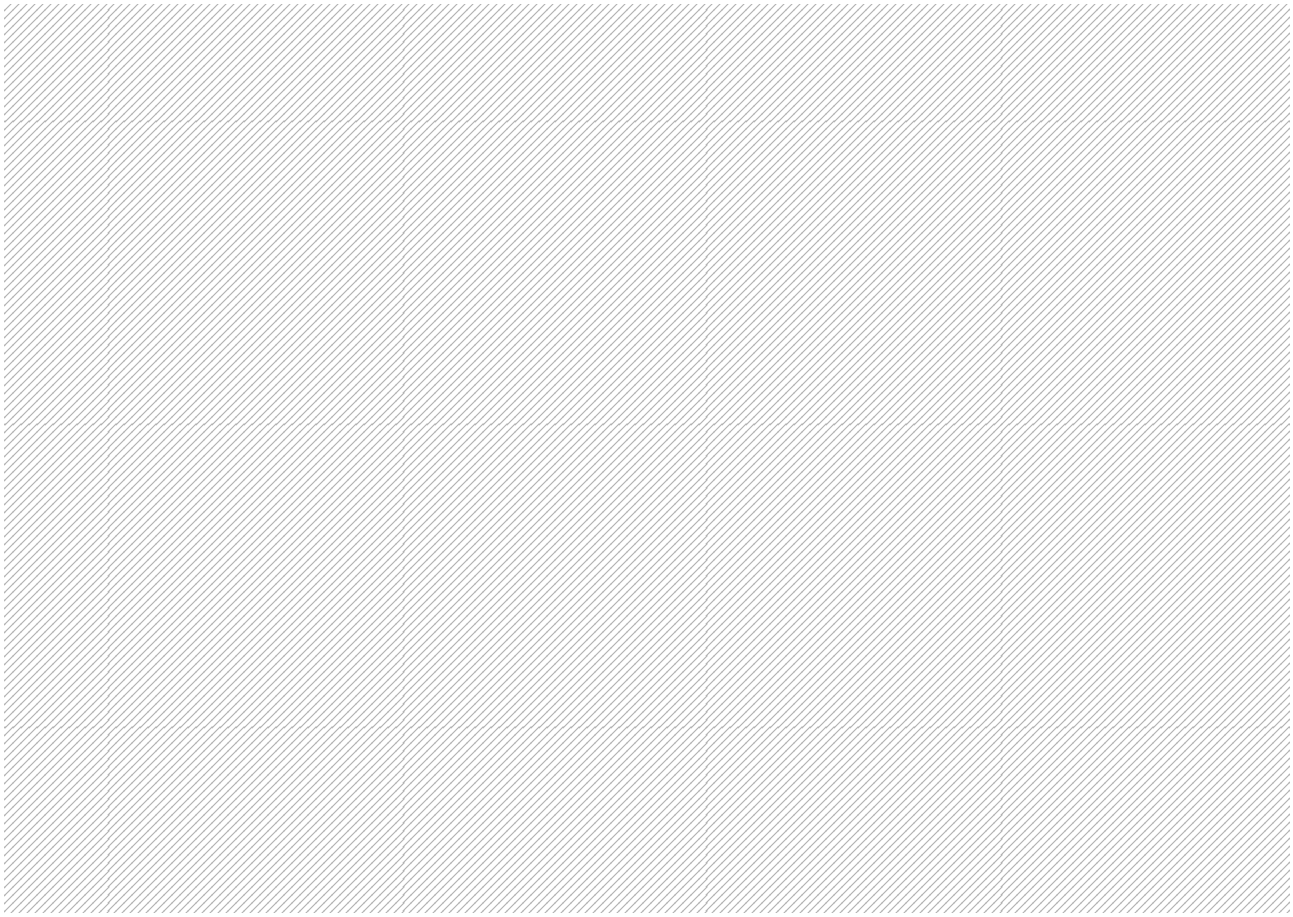
bare-cross/ ..... no frame, tree still breaks
├── src/
│ ├── a.py
│ ├── b.py
│ ├── c.py
│ ├── d.py
│ ├── e.py
│ ├── f.py
│ ├── g.py
│ ├── h.py
│ └── i.py
├── tests/
│ │

```
    ├── t1.py
    ├── t2.py
    ├── t3.py
    └── t4.py
├── README.md
└── LICENSE
```

10.6 Bare tree with `tree break at` pull-back

Even without a frame, `tree break at` still applies: the tree extension columns retract from the page edge by the configured distance on each side of the break. Setting it asymmetrically lets the bottom of the first piece and the top of the second piece pull back by different amounts.

```
\begin{dirtreex}[
  box=false,
  pagebreak={true, tree break at={2em, 4em}}]
  \dir{bare-pullback}{first-piece-bottom=2em, next-piece-top=4em}{
    \dir{src}{}{
      \file{a.py}{}\file{b.py}{}\file{c.py}{}
      \file{d.py}{}\file{e.py}{}\file{f.py}{}
      \file{g.py}{}\file{h.py}{}
    }
    \dir{docs}{}{
      \file{x.md}{}\file{y.md}{}\file{z.md}{}
    }
  }
\end{dirtreex}
```



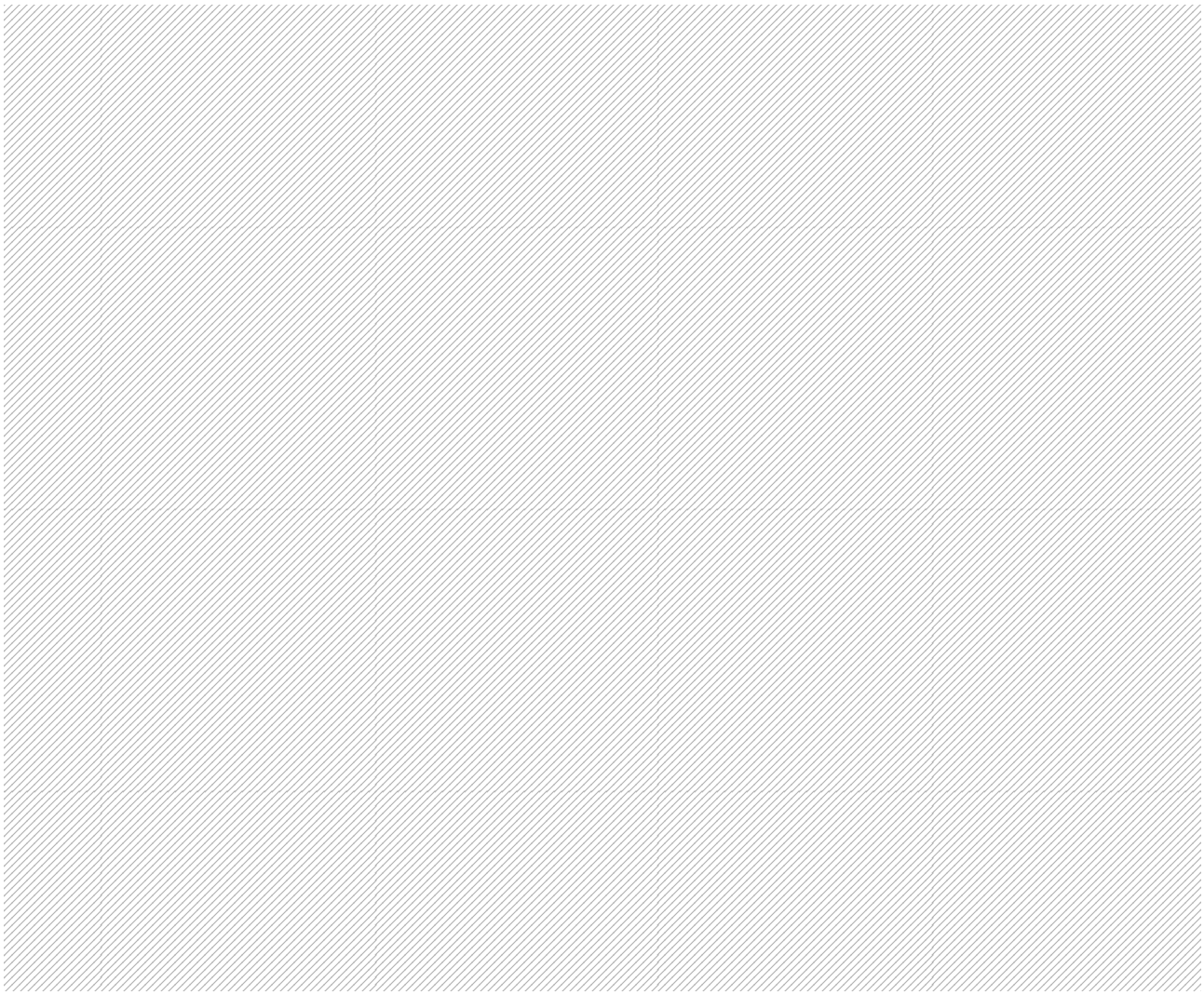
bare-pullback/ ..... first-piece-bottom=2em, next-piece-top=4em
├── src/
│   ├── a.py
│   ├── b.py
│   ├── c.py
│   ├── d.py
│   ├── e.py

```
    ├── f.py
    ├── g.py
    └── h.py
└── docs/
    ├── x.md
    ├── y.md
    └── z.md
```

10.7 `pagebreak=false` (tree spills past the page)

With `pagebreak=false` the frame refuses to break; a tall tree simply overflows past the page boundary (it is never truncated). This example uses a small filler so the spillover is visible.

```
\begin{dirtreex}[
  pagebreak={false},
  box={true, border color=black!50,
       background color=black!3, margin=6pt}]
  \dir{no-break}{overflows past page bottom}{
    \file{a.txt}{}\file{b.txt}{}\file{c.txt}{}
    \file{d.txt}{}\file{e.txt}{}\file{f.txt}{}
    \file{g.txt}{}\file{h.txt}{}\file{i.txt}{}
    \file{j.txt}{}
  }
\end{dirtreex}
```



```
no-break/.....................................................................overflows past page bottom
├── a.txt
├── b.txt
├── c.txt
├── d.txt
├── e.txt
├── f.txt
├── g.txt
├── h.txt
├── i.txt
└── j.txt
```

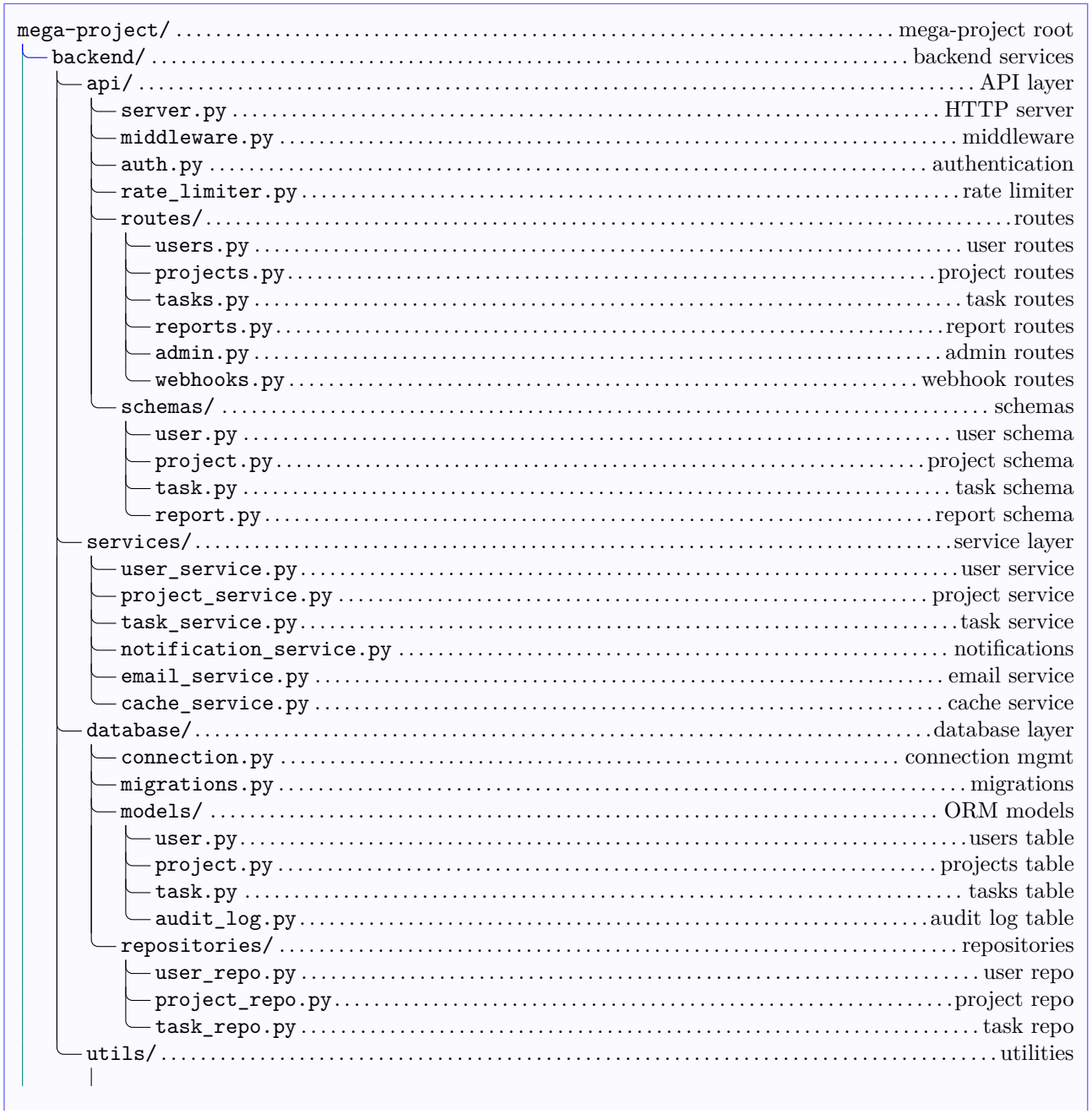
11 Multi-page mega-project

A realistic tree large enough to span three or more pages. It exercises every cross-page mechanic at once: per-branch colours, nested directory depth, mixed leaf and subtree siblings, and the `|` extension columns continuing across two consecutive breaks. Each top-level branch carries its own `line color` override, which the package threads through every active column on every continuation page.

```
\begin{dirtreex}[
  fontsize=\small, elbow radius=3pt, line color=black,
  box={true, corners=0pt, border color=blue!60,
       border width=0.5pt, background color=blue!2,
       margin=6pt}]
  \dir{mega-project}{mega-project root}{
    \dir[line color=blue]{backend}{backend services}{
      \dir{api}{API layer}{
        \file{server.py}{HTTP server}
        \file{middleware.py}{middleware}
        \file{auth.py}{authentication}
        \file{rate\_limiter.py}{rate limiter}
        \dir{routes}{routes}{
          \file{users.py}{user routes}
          \file{projects.py}{project routes}
          \file{tasks.py}{task routes}
          \file{reports.py}{report routes}
          \file{admin.py}{admin routes}
          \file{webhooks.py}{webhook routes}
        }
        \dir{schemas}{schemas}{
          \file{user.py}{user schema}
          \file{project.py}{project schema}
          \file{task.py}{task schema}
          \file{report.py}{report schema}
        }
      }
      \dir{services}{service layer}{
        \file{user\_service.py}{user service}
        \file{project\_service.py}{project service}
        \file{task\_service.py}{task service}
        \file{notification\_service.py}{notifications}
        \file{email\_service.py}{email service}
        \file{cache\_service.py}{cache service}
      }
      \dir{database}{database layer}{
        \file{connection.py}{connection mgmt}
        \file{migrations.py}{migrations}
        \dir{models}{ORM models}{
          \file{user.py}{users table}
          \file{project.py}{projects table}
          \file{task.py}{tasks table}
          \file{audit\_log.py}{audit log table}
        }
        \dir{repositories}{repositories}{
          \file{user\_repo.py}{user repo}
          \file{project\_repo.py}{project repo}
          \file{task\_repo.py}{task repo}
        }
      }
      \dir{utils}{utilities}{
        \file{logger.py}{logging}
        \file{config.py}{config}
        \file{helpers.py}{helpers}
        \file{validators.py}{validators}
        \file{crypto.py}{crypto utils}
      }
```

```
}
\dir[line color=teal]{frontend}{frontend app}{
  \dir{src}{source}{
    \dir{components}{components}{
      \dir{common}{common}{
        \file{Button.tsx}{button}
        \file{Input.tsx}{input}
        \file{Modal.tsx}{modal}
        \file{Table.tsx}{table}
        \file{Pagination.tsx}{pagination}
        \file{Loading.tsx}{loading}
        \file{ErrorBoundary.tsx}{error boundary}
      }
      \dir{layout}{layout}{
        \file{Header.tsx}{header}
        \file{Sidebar.tsx}{sidebar}
        \file{Footer.tsx}{footer}
        \file{MainContent.tsx}{main content}
      }
      \dir{pages}{pages}{
        \file{Dashboard.tsx}{dashboard}
        \file{ProjectList.tsx}{project list}
        \file{ProjectDetail.tsx}{project detail}
        \file{TaskBoard.tsx}{task board}
        \file{UserProfile.tsx}{user profile}
        \file{Settings.tsx}{settings}
        \file{Login.tsx}{login}
        \file{Register.tsx}{register}
      }
    }
    \dir{hooks}{custom hooks}{
      \file{useAuth.ts}{auth hook}
      \file{useApi.ts}{api hook}
      \file{useWebSocket.ts}{websocket hook}
      \file{useLocalStorage.ts}{local storage hook}
    }
    \dir{stores}{stores}{
      \file{authStore.ts}{auth state}
      \file{projectStore.ts}{project state}
      \file{uiStore.ts}{ui state}
    }
    \dir{utils}{utilities}{
      \file{api.ts}{api client}
      \file{format.ts}{format}
      \file{validation.ts}{form validation}
    }
    \file{App.tsx}{app entry}
    \file{index.tsx}{render entry}
    \file{router.tsx}{router config}
  }
  \file{package.json}{dependencies}
  \file{tsconfig.json}{typescript config}
  \file{vite.config.ts}{vite config}
}
\dir[line color=orange]{devops}{devops \& deployment}{
  \dir{docker}{docker config}{
    \file{Dockerfile.backend}{backend image}
    \file{Dockerfile.frontend}{frontend image}
    \file{docker-compose.yml}{compose file}
    \file{docker-compose.dev.yml}{dev compose}
  }
  \dir[line color=green]{k8s}{k8s config}{
    \file{deployment.yaml}{deployment}
    \file{service.yaml}{service}
    \file{ingress.yaml}{ingress}
    \file{configmap.yaml}{configmap}
```

```
        \file{secret.yaml}{secret}
        \file{hpa.yaml}{horizontal autoscaler}
      }
      \dir{ci}{CI}{
        \file{.gitlab-ci.yml}{gitlab ci}
        \file{Jenkinsfile}{jenkins pipeline}
      }
      \dir{monitoring}{monitoring}{
        \file{prometheus.yml}{prometheus config}
        \file{grafana-dashboard.json}{grafana dashboard}
        \file{alertmanager.yml}{alert rules}
      }
    }
    \file{README.md}{project readme}
    \file{LICENSE}{license}
    \file{.gitignore}{git ignore rules}
    \file{Makefile}{build script}
  }
\end{dirtreex}
```



```
        ├── logger.py ..... logging
        ├── config.py ..... config
        ├── helpers.py ..... helpers
        ├── validators.py ..... validators
        └── crypto.py ..... crypto utils
└── frontend/ ..... frontend app
    ├── src/ ..... source
    │   ├── components/ ..... components
    │   │   ├── common/ ..... common
    │   │   │   ├── Button.tsx ..... button
    │   │   │   ├── Input.tsx ..... input
    │   │   │   ├── Modal.tsx ..... modal
    │   │   │   ├── Table.tsx ..... table
    │   │   │   ├── Pagination.tsx ..... pagination
    │   │   │   ├── Loading.tsx ..... loading
    │   │   │   └── ErrorBoundary.tsx ..... error boundary
    │   │   ├── layout/ ..... layout
    │   │   │   ├── Header.tsx ..... header
    │   │   │   ├── Sidebar.tsx ..... sidebar
    │   │   │   ├── Footer.tsx ..... footer
    │   │   │   └── MainContent.tsx ..... main content
    │   │   └── pages/ ..... pages
    │   │       ├── Dashboard.tsx ..... dashboard
    │   │       ├── ProjectList.tsx ..... project list
    │   │       ├── ProjectDetail.tsx ..... project detail
    │   │       ├── TaskBoard.tsx ..... task board
    │   │       ├── UserProfile.tsx ..... user profile
    │   │       ├── Settings.tsx ..... settings
    │   │       ├── Login.tsx ..... login
    │   │       └── Register.tsx ..... register
    │   ├── hooks/ ..... custom hooks
    │   │   ├── useAuth.ts ..... auth hook
    │   │   ├── useApi.ts ..... api hook
    │   │   ├── useWebSocket.ts ..... websocket hook
    │   │   └── useLocalStorage.ts ..... local storage hook
    │   ├── stores/ ..... stores
    │   │   ├── authStore.ts ..... auth state
    │   │   ├── projectStore.ts ..... project state
    │   │   └── uiStore.ts ..... ui state
    │   ├── utils/ ..... utilities
    │   │   ├── api.ts ..... api client
    │   │   ├── format.ts ..... format
    │   │   └── validation.ts ..... form validation
    │   ├── App.tsx ..... app entry
    │   ├── index.tsx ..... render entry
    │   └── router.tsx ..... router config
    ├── package.json ..... dependencies
    ├── tsconfig.json ..... typescript config
    └── vite.config.ts ..... vite config
└── devops/ ..... devops & deployment
    ├── docker/ ..... docker config
    │   ├── Dockerfile.backend ..... backend image
    │   ├── Dockerfile.frontend ..... frontend image
    │   ├── docker-compose.yml ..... compose file
    │   └── docker-compose.dev.yml ..... dev compose
    └── k8s/ ..... k8s config
        ├── deployment.yaml ..... deployment
        │
```

```
        ├── service.yaml ..... service
        ├── ingress.yaml ..... ingress
        ├── configmap.yaml ..... configmap
        ├── secret.yaml ..... secret
        └── hpa.yaml ..... horizontal autoscaler
    ├── ci/ ..... CI
    │   ├── .gitlab-ci.yml ..... gitlab ci
    │   └── Jenkinsfile ..... jenkins pipeline
    └── monitoring/ ..... monitoring
        ├── prometheus.yml ..... prometheus config
        ├── grafana-dashboard.json ..... grafana dashboard
        └── alertmanager.yml ..... alert rules
├── README.md ..... project readme
├── LICENSE ..... license
├── .gitignore ..... git ignore rules
└── Makefile ..... build script
```

12 Verbatim names

These examples use the verbatim variants `\verbdir` / `\verbfile` / `\verbarchive`. The path argument is read under xparse's `v` specifier; catcodes inside the chosen delimiters are set to 12 ("other"), so common dangerous bytes survive as literal glyphs.

12.1 Top-level verbatim file

A single `\verbfile` with an underscore and a dollar sign in the name:

```
\begin{dirtreex}
  \verbfile|app_$STAGE.toml|{environment-templated config}
\end{dirtreex}
```

app_$STAGE.toml ..... environment-templated config

12.2 Nested verbdir containing verbfile

A `\verbfile` or `\verbdir` nested inside another `\verbdir`'s children block works because the children block is a live TeX group — the inner `v` argument reads its delimiter pair directly from the source, with no intervening tokenisation step.

```
\begin{dirtreex}
  \verbdir|src/data_$|{data root with dollar in name}{%
    \verbfile|users_2026-01.parquet|{snapshot}%
    \verbfile|conf{prod}.yaml|{production config}%
  }
\end{dirtreex}
```

src/data_$/ ..... data root with dollar in name
├── users_2026-01.parquet ..... snapshot
└── conf{prod}.yaml ..... production config

12.3 Verbatim archive name

`\verbarchive` is the verbatim sibling of `\archive` — container-shaped (children block, no trailing `/`), but with the path read at catcode 12. Useful when an archive's filename contains shell metacharacters or template placeholders that would otherwise need escaping.

```
\begin{dirtreex}
  \verbarchive|build_$STAGE.tar.gz|{templated archive name}{%
    \verbfile|manifest_$.json|{}%
    \verbfile|conf{prod}.yaml|{}%
  }
\end{dirtreex}
```

```
build_$STAGE.tar.gz ..... templated archive name
├── manifest_$.json
└── conf{prod}.yaml
```

12.4 Mixed normal and verbatim siblings

`\dir` / `\file` / `\archive` (normal-catcode names) and `\verbdir` / `\verbfile` / `\verbarchive` (verbatim names) coexist freely inside the same children block.

```
\begin{dirtreex}
  \dir{project}{root}{%
    \dir{src}{source}{%
      \file{main.py}{}%
      \verbfile|loader_$.py|{verbatim sibling}%
    }%
    \verbdir|cfg_%|{verbatim dir sibling}{%
      \file{default.yaml}{}%
    }%
  }
\end{dirtreex}
```

```
project/ ..... root
├── src/ ..... source
│   ├── main.py
│   └── loader_$.py ..... verbatim sibling
└── cfg_%/ ..... verbatim dir sibling
    └── default.yaml
```

12.5 Pipe (`|`) in the path — alternative delimiter

The default delimiter shown above is `|`, but *any* non-letter, non-space character may be chosen. Pick one that does not appear in the path. Below, `+` and `/` are used as delimiters so a literal `|` survives in the name:

```
\begin{dirtreex}
  \dir{shell-pipelines}{names that contain a literal |}{%
    \verbfile+a|b|c+{+ as delimiter}%
    \verbfile/ps aux | grep python/{/ as delimiter}%
    \verbdir!logs|archive!{! as delimiter}{%
      \verbfile=out|err.log={= as delimiter, file inside}%
    }%
  }
\end{dirtreex}
```

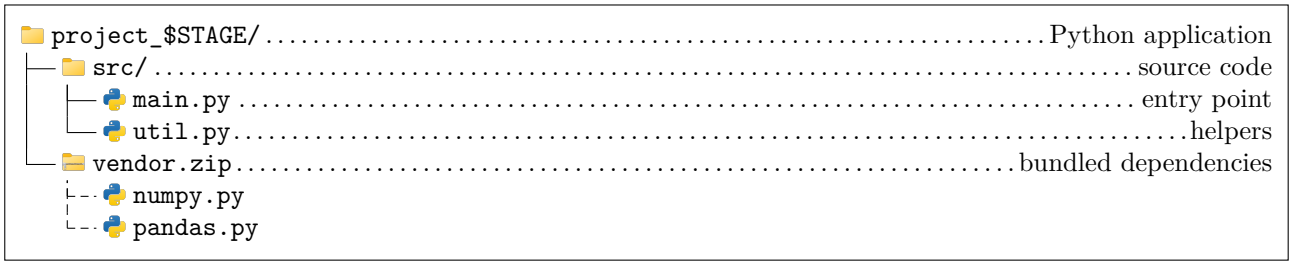
```
shell-pipelines/ ..... names that contain a literal |
├── a|b|c ..... + as delimiter
├── ps aux | grep python ..... / as delimiter
└── logs|archive/ ..... ! as delimiter
    └── out|err.log ..... = as delimiter, file inside
```

A delimiter *collision* corrupts silently rather than erroring: `\verbfile|a|b|{c}` takes `|a|` as the name and `b` as the comment, leaving `|{c}` as stray tokens. Pick a delimiter that never appears in the names you intend to write.

13 Icon dispatch with \DirtreexFormatName

The public hook \DirtreexFormatName controls how each entry's name is typeset. Override it with \renewcommand and dispatch on the entry's t slot (0 for files, 1 for directories, 2 for archives) to attach per-type icons. The example below pulls three PNG glyphs from the assets/ directory shipped with this repository and renders one in front of every entry name:

```
\newcommand{\dteicon}[1]{%
  \raisebox{-.15em}{\includegraphics[height=.9em]{assets/#1}}%
  \kern0.5\fontdimen2\font}
\renewcommand{\DirtreexFormatName}[1]{%
  \ifcase\DirtreexGetField{#1}{t}%
    \dteicon{python.png}\DirtreexGetField{#1}{n}%                % t=0 file
  \or
    \dteicon{directory.png}\DirtreexGetField{#1}{n}/%            % t=1 directory
  \or
    \dteicon{zip_file.png}\DirtreexGetField{#1}{n}%              % t=2 archive
  \fi
}
\begin{dirtreex}
  \verbdir|project_$STAGE|{Python application}{
    \dir{src}{source code}{
      \file{main.py}{entry point}
      \file{util.py}{helpers}
    }
    \archive{vendor.zip}{bundled dependencies}{
      \file[line style = densely dashed]{numpy.py}{}
      \file[line style = densely dashed]{pandas.py}{}
    }
  }
\end{dirtreex}
```



The \begingroup/\endgroup wrapper scopes both \newcommand and \renewcommand, so the override does not leak into anything compiled after this section. To apply the icons document-wide, place the two definitions in your preamble instead.