Prefetch

```rust
#![allow(clippy::unwrap_used, clippy::expect_used)]
use std::path::PathBuf;
use std::fs;
use std::path::Path;
use chrono::{DateTime, Utc};

/// based on
https://docs.rs/crate/prefetch-core/0.1.0/source/examples/pf_dump.rs


fn filetime_to_datetime(ft: i64) -> DateTime<Utc> {
    // FILETIME epoch (1601-01-01) -> Unix epoch (1970-01-01)
    const EPOCH_DIFF_100NS: i64 = 116_444_736_000_000_000;

    let unix_100ns = ft - EPOCH_DIFF_100NS;
    let secs = unix_100ns / 10_000_000;
    let nanos = ((unix_100ns % 10_000_000) * 100) as u32;

    DateTime::from_timestamp(secs, nanos)
        .expect("invalid timestamp")
}

fn csv_escape(s: &str) -> String {
    let mut out = String::new();
    out.push('"');
    for c in s.chars() {
        match c {
            '"' => out.push_str("\"\""),
            _ => out.push(c),
        }
    }
    out.push('"');
    out
}

fn main() {
    let mut root = PathBuf::from(env!("CARGO_MANIFEST_DIR"));
    root.pop();
    let out_dir = std::env::args()
        .nth(1)
        .unwrap_or_else(|| "/tmp/pf_scca".to_string());
    std::fs::create_dir_all(&out_dir).expect("mkdir out_dir");

    let dir = Path::new("/media/data/IR/case260618/prefetch");

     let files: Vec<String> = fs::read_dir(dir)
        .unwrap()
```

```
        .filter_map(Result::ok)
        .map(|e| e.path())
        .filter(|p| {
            p.extension()
                .and_then(|ext| ext.to_str())
                .map(|ext| ext == "pf")
                .unwrap_or(false)
        })
        .map(|p| p.to_string_lossy().to_string())
        .collect();

        let mut wtr = csv::Writer::from_writer(std::io::stdout());

        // header
        wtr.write_record([
            "file",
            "scca_len",
            "version",
            "executable",
            "run_count",
            "last_run_times",
            "filenames_count",
            "filenames",
            "volumes",
        ]).unwrap();

    for name in files {
        let p = PathBuf::from(&name);

        let raw = std::fs::read(&p).expect("read fixture");
        let scca = prefetch_core::decompress(&raw).expect("decompress");

        std::fs::write(PathBuf::from(&out_dir).join(format!("{name}.scca")),
&scca)
            .expect("write scca");

        let info = prefetch_core::parse(&raw).expect("parse");

        let filenames = info
            .filenames
            .iter()
            .map(|f| escape(f))
            .collect::<Vec<_>>()
            .join(";");

        let volumes = info
            .volumes
            .iter()
            .map(|v| {
                format!(
                    "{{serial:{},device_path:{},creation_time:{}}}",
```

```
                    v.serial,
                    escape(&v.device_path),
                    filetime_to_datetime(v.creation_time)
                )
            })
            .collect::<Vec<_>>()
            .join(";");

        let last_runs = info
            .last_run_times
            .iter()
            .map(|t| filetime_to_datetime(*t).to_string())
            .collect::<Vec<_>>()
            .join(";");

        wtr.write_record(&[
        name,
        scca.len().to_string(),
        info.version.to_string(),
        info.executable,
        info.run_count.to_string(),
        last_runs,
        info.filenames.len().to_string(),
        filenames,
        volumes,
        ]).unwrap();
    }
}

fn escape(s: &str) -> String {
    s.replace('\\', "\\\\")
}
```