# PPI/1.0 — Protocolo de Proyecto Integrado (para IPC)

### Curso Integrador (CI0123) — Primera Etapa

## 1) Propósito y alcance

- Objetivo: estandarizar el diálogo Cliente ↔ Tenedor ↔ Servidor de Figuras en la simulación, para listar, consultar manifiestos y obtener partes (ASCII) de figuras.
- Ámbito: simulación sin sockets, de momento (IPC local); no define seguridad ni autenticación.
- Inspiración: sintaxis “HTTP-like” (líneas y encabezados), pero con nombres propios y reglas simplificadas.

## 2) Transporte (IPC) y codificación

- Transporte: cualquier IPC local: pipes, FIFOs, colas POSIX (mq_*), o memoria compartida + semáforos.
- Codificación: UTF-8 en todo el mensaje.
- Conexión lógica: persistente hasta que una parte envíe 'Connection: close' o termine el proceso/hilo.

## 3) Framing del mensaje (formato general)

Todo mensaje consta de:

```
<start-line>\r\n
<encabezado-1>: <valor>\r\n
<encabezado-2>: <valor>\r\n
...
\r\n
<body...>            ; opcional, longitud fija
```

- Delimitación de encabezados: secuencia en blanco \r\n (CRLF doble).
- Tamaño del cuerpo: obligatorio con 'Len: ' cuando N > 0. Si no hay cuerpo, use 'Len: 0' u omita 'Len'.
- Límites recomendados: encabezados ≤ 4 KiB; Len ≤ 64 KiB (ajustable).

## 4) Start-line y encabezados estándar

Start-line de solicitud

```
PPI/1.0 REQ <rid> <VERB> <RUTA>
- <rid>: request id (entero o token) para correlacionar respuestas.
- <VERB>: comandos del §5.
- <RUTA>: recurso (p. ej., /figures, /figures/<name>/parts/<part>).
```

Start-line de respuesta

```
PPI/1.0 RES <rid> <código> <razón>
- <código>: ver §6 (200, 404, …).
- <razón>: texto corto (OK, NOT_FOUND, …).
```

Encabezados estándar (propios)

```
- Len: <N> — longitud del cuerpo en bytes (cuando hay cuerpo).
- Type: <mime-sencillo> — tipo lógico del cuerpo: text/list, text/parts, text/plain, text
- Connection: keep-alive | close — cierre lógico.
- Opcionales: From: CLIENT|FORK|SERVER, X-Req-Id, X-Forwarded-For.
```

## 5) Comandos (verbs) y rutas

5.1 LIST — listar figuras
Solicitud:

```
PPI/1.0 REQ 1 LIST /figures
Connection: keep-alive
```

Respuesta (200 OK):

```
PPI/1.0 RES 1 200 OK
Type: text/list
Len: 18

ballena
pez
tortuga
```

5.2 MANIFEST — ver partes de una figura
Solicitud:

```
PPI/1.0 REQ 2 MANIFEST /figures/ballena/manifest
Connection: keep-alive
```

Respuesta (200 OK):

```
PPI/1.0 RES 2 200 OK
Type: text/parts
Len: 16

head
body
tail
```

5.3 PART — obtener una parte concreta
Solicitud:

```
PPI/1.0 REQ 3 PART /figures/ballena/parts/head
Connection: keep-alive
```

Respuesta (200 OK):

```
PPI/1.0 RES 3 200 OK
Type: text/plain
Len: 23

   __/\\__   (ASCII)
```

## 6) Códigos de estado

200 OK — éxito
204 NO_CONTENT — sin cuerpo
400 BAD_REQUEST — sintaxis/encabezados inválidos
404 NOT_FOUND — figura/parte inexistente
409 CONFLICT — inconsistencias
413 TOO_LARGE — excede límites
500 ERROR — fallo interno

## 7) Máquina de estados (sencilla)

[READY] --(LIST/MANIFEST/PART)--> [READY]
[READY] --(BYE)----------------> [CLOSE]

## 8) Reglas de parsing (receptor)

1) Leer encabezados hasta \r\n\r\n.
2) Validar start-line: 'PPI/1.0 REQ/RES ...'.
3) Parsear encabezados (Type, Len, Connection, ...).
4) Leer el cuerpo: si Len: N, leer exactamente N bytes.
5) Entregar al handler según VERB.
6) Logging: imprimir 'ACTOR -> DEST: start-line' y 'ACTOR <- SRC: RES ...'.

## 9) Mapeo a IPC

- Pipes/FIFOs: flujo de bytes. Usa el framing (\r\n\r\n + Len) para delimitar.
- Colas POSIX: si mq_msgsize alcanza, manda mensaje completo; si no, usa encabezado de longitud propio.
- Memoria compartida: buffer circular; cada publicación es 'headers + \r\n\r\n + body'; sincroniza con semáforos.

## 10) Errores y resiliencia

- Len > límite → 413 TOO_LARGE.
- Len ausente cuando se esperaba cuerpo → 400 BAD_REQUEST.
- Figura/parte inexistente → 404 NOT_FOUND.
- EINTR → reintentar; en no-bloqueante, devolver 0 y reintentar en el reactor.

## 11) Formatos de cuerpo (simplificados)

- text/list y text/parts: una entrada por línea ('\n').
- text/plain: blob ASCII tal cual.
- text/json: si el equipo lo prefiere (no requerido).

## 12) Ejemplo de sesión (con Tenedor reenviando)

Cliente → Tenedor

```
PPI/1.0 REQ 1 LIST /figures
From: CLIENT
Connection: keep-alive
```

Tenedor → Servidor

```
PPI/1.0 REQ 1 LIST /figures
From: FORK
X-Forwarded-For: CLIENT
Connection: keep-alive
```

Servidor → Tenedor

```
PPI/1.0 RES 1 200 OK
Type: text/list
Len: 12

pez
```

```
ballena
```

Tenedor → Cliente (reenvío)

```
PPI/1.0 RES 1 200 OK
Type: text/list
Len: 12

pez
ballena
```

Cierre

```
PPI/1.0 REQ 99 BYE /control/bye
Connection: close
→ RES 99 200 BYE
```

## 13) Extensibilidad y criterios de aceptación

- Encabezados no reconocidos se ignoran (prefijo X- recomendado).
- Nuevos verbs no rompen compatibilidad si respetan framing y Len.
- En demo: rótulos visibles; LIST/MANIFEST/PART/BYE operan; receptor lee exactamente Len; funciona con IPC; Makefile y ejecución en el entorno del curso.