

# PandOS+

Fase 2

**Cheikh Ibrahim · Zaid**

Matricola: 0000974909

**Lee · Qun Hao Henry**

Matricola: 0000990259

**Xia · Tian Cheng**

Matricola: 0000975129

**Paris · Manuel**

Matricola: 0000997526

Anno accademico

2021 — 2022

Corso di Sistemi Operativi

Alma Mater Studiorum · Università di Bologna

# 1 Introduzione

La seconda fase del progetto PandOS+ consiste nell'implementazione del livello 3 dell'architettura astratta di un sistema operativo proposta da Dijkstra. In particolare è necessario implementare le funzionalità del kernel per l'inizializzazione del sistema, la schedulazione dei processi e la gestione delle eccezioni.

## 1.1 Organizzazione dei file

I file coinvolti per l'implementazione delle specifiche sono i seguenti:

|                         |  |
|-------------------------|--|
| <code>pcb</code>        | È stata aggiunta la gestione dei <code>pid</code> dei processi.  |
| <code>asl</code>        | Sono state aggiunte le operazioni P/V sui semafori.  |
| <code>initial</code>    | Gestisce la fase di inizializzazione del sistema e fornisce alcune funzioni ausiliare di utilità generale. |
| <code>scheduler</code>  | Implementa lo scheduler.   |
| <code>exceptions</code> | Reindirizza le eccezioni ai relativi gestori. Implementa il gestore delle system call.                     |
| <code>interrupts</code> | Implementa il gestore delle interrupt.   |
| <code>utilities</code>  | Contiene l'implementazione di <code>memcpy</code> .  |

## 2 Gestione dei pid

I pid vengono generati in ordine crescente a partire da 1.

La mappatura dei pid validi in un dato istante avviene tramite una lista bidirezionale implementata attraverso un nuovo campo aggiunto della struttura `pcb_t`. Tale lista è ordinata in senso crescente per migliorare le prestazioni di ricerca e permette l'inserimento in tempo costante per il primo "ciclo di pid" (dopo un wraparound non è garantita questa proprietà).

### 2.1 Possibili miglioramenti

Sono possibili alternative a tale implementazione utilizzando strutture dati differenti (es. alberi di ricerca) che permettono di ottenere prestazioni diverse.

La scelta di utilizzare una lista ordinata è stata fatta considerando un trade-off tra semplicità di implementazione e prestazioni che rimangono valide in quanto il numero massimo di processi attivi ha un limite superiore molto contenuto.

## 3 Gestione dei semafori per i device

### 3.1 Interval timer

L'interval timer ha un semaforo dedicato e il tempo caricato ad ogni ciclo viene calcolato in base alla seguente formula:

$$\text{time}_{IT} = \text{PSECOND} - (\text{curr\_time} \% \text{PSECOND}) \quad (1)$$

Dove `PSECOND` è la durata dell'interval timer e `curr_time` è il valore del `TOD`.

In questo modo, il timer viene impostato facendo riferimento al momento di inizio del `TOD`, permettendo di non accumulare eventuali ritardi causati dalla gestione dell'interrupt.

### 3.2 Device di I/O

I semafori associati ai device di I/O sono memorizzati in un vettore la cui mappatura tra indice e device segue la seguente formula:

$$i = \frac{\text{dev\_reg\_address} - \text{dev\_reg\_start\_address}}{\text{dev\_reg\_size}} \quad (2)$$

Dove:

- `dev_reg_address` è il parametro che indica l'indirizzo del device register associato al device per cui si vuole trovare il semaforo <sup>1</sup>.
- `dev_reg_start_address` = 0x10000054 è l'indirizzo da cui iniziano i device register.
- `dev_reg_size` = 0x10 è la dimensione dei device register (4 word).

Quindi al primo indice del vettore si trova il semaforo associato al device di I/O con priorità maggiore (disco 1) e all'ultimo quello con priorità minore (trasmissione terminale 7).

## 4 Gestione del tempo di CPU

Il calcolo del tempo di CPU per ciascun processo utilizza, come da specifiche, il meccanismo di temporizzazione fornito dal `TOD`.

La politica di accumulo prevede di contabilizzare, per ciascun processo, il tempo di esecuzione effettivo e il tempo impiegato per eseguire le system call (che sono direttamente invocate dal processo). Non viene invece considerato il tempo di gestione degli interrupt, durante il quale viene disabilitato il `PLT`, in quanto non sono causate in modo diretto dal processo.

---

<sup>1</sup>Durante l'elaborazione, viene fornito l'indirizzo del campo *command* del device register. A partire da tale indirizzo è possibile risalire all'indirizzo di inizio del device register (bisogna gestire il fatto che i terminali hanno due campi *command*).