GESTIONE PROCESSI

STATI PROCESSI → condizioni in cui può trovarsi un processo: → **ATTESA**, per continuare a eseguire deve verificarsi un evento;
→ il **processo corrente** è quello in esecuzione;
→ **PRONTO**, può essere schedulato;

CONTESTO DI UN PROCESSO

→ se il processo effettua una SYS-CALL una funzione dell'S.O. viene attivata per eseg. il servizio:
- **U** → esec. standard
- **S** → S.O. in esecuzione nel contesto di tale processo (o per SYS-CALL o per interrupt)

ABBANDONO ESEC DI UN PROC

↳ avviene in due casi: → **attesa evento** (→ risposta da device non dispo o da altro proc), il processo passa quindi in stato di wait;
→ **serv di sistema**, ovvero syscall chiamata da lui;
→ **sist preemptable**, passa allo stato di pronto;

→ lo **SCHEDULER** si occupa di effettuare questi passaggi realizzando **politica di scheduling** ed eseguendo il **context switch**;
↳ svolto dalla funzione schedule()
↳ gestisce anche una struttura dati fondamentale (per la CPU) ovvero **runqueue**
→ **RB**, lista dei vari descrittori di processi;
→ **CURR**, puntatore al descrittore del processo corrente;

→ **sPila**: parte di pila contenente il contesto HW di un processo sospeso durante una chiamata di servizio;
↳ passaggio da uPila a sPila con salvataggio dei valori SSP e USP
↳ per fare ciò sono presenti due campi nel Desc. di Proc: **sp0**, contiene indr base sPila; **sp**, contiene val. sp nel momento di interruzione;

→ **GESTIONE SSP e USP NEL CONT. SWITCH:**
- proc P in esec in modo U → sPila vuota, SSP → sp0;
- CPU passa a S → USP salva SP corrente per il ritorno al proc. interrotto (→ a mod U);
- **se avviene cont. switch durante modo S?** salva USP su sPila di P → SP nel campo sp
  ↳ quando P riprende esec ricarico SP da descrittore (campo sp), USP ricaric. da sPila, SSP ricaric. da sp0

↳ questo sistema risulta semplificato da una serie di prob. come gestione ① interrupt, passaggio ② ATTESA→PRONTO, preemption ③;

① **gestione interrupt** → può verificarsi in 3 casi: durante proc in mode U, durante SYS-CALL, durante una int-serv-routine.
↳ con priorità inferiore
↳ le interrupt sono eseguite **nel contesto** del processo che interrompono (non viene quindi interrotto)
infatti se la interrupt viene chiamata da un evento E con proc P in attesa, quest'ultimo passa da ATTESA a PRONTO (→ riparte dopo l'interrupt)

② **gestione stato di attesa** (waitqueue) → 3 casi di attesa: completam. operaz. I/O, lock da mutex (o sem), scadenza timeout.
↳ lista di puntatori a descrittori dei processi in attesa **di un certo evento**
↳ il suo indirizzo costituisce l'ide dell'evento;
creaz (static): DECLARE_WAIT_QUEUE_HEAD nome_coda
↳ esistono priorità di attesa (esclusiva e non) con flag che contrassegna questi processi
(↳ periferiche)

SEGNALI → avviso asincrono da proc. a S.O. o ad un altro processo (→ es la kill), identif con numero tra 1 e 31 e da un nome

↳ quando arriva a un proc. o esegue tram funzione apposita oppure esegue 'default signal handler'

↳ esistono invece due segnali non intercettabili dal proc. che esegue:
- SIGKILL;
- SIGSTOP;

↳ solo se proc in modo U. Se il processo è in modo S il sign. viene processato al ritorno in modo U. Il ritorno dal tipo di attesa (se è preemptable o meno).

↳ se si il proc. verificherà se può essere risvegliato senza che l'evento che attendeva si verifichi.

es. wait_event(...,...), wait_event_killable(...,..), wait_event_interruptible(...,...)

↳ funzioni di attesa per processi

2.1 risveglio di un processo → wake_up(wait_queue_head_t* wq)

risveglia tutti i task in attesa non escl. e solo 1 in esclusiva

porta in stato di pronto

da waitqueue → runqueue

↳ se un task arriva in runqueue e ha maggiori diritti di altri in via di esec, wake_up pone a 1 il flag TIF_NEED_RESCHED

sembra violare la non-preemptable del nucleo ma in realtà si tratta solo di eseguirla appena prima della IRET (o SYSRET) al modo U ('se necessario' = se devo eseg. un altro prog. con priorità)

3 attesa di un timeout → risveglio alla fine di una scadenza di tempo di esec. (vengono contati i clock)

↳ i timer vengono controllati periodicamente per una questione di efficienza;

RIASSUNTO →