LAB 4
WAITPID

PROGRAMMAZIONE DI SISTEMA (JZ-ZZ)

2023/24

ANDREA PORTALURI

OBIETTIVI DEL LABORATORIO

- Implementazione della system call waitpid
- (opzionale) implementazione delle system calls getpid e fork

ATTENZIONE: per questo laboratorio è richiesto il completamento dei laboratori 2 e 3, aver compreso il flow di implmementazione di una system call (in particolare, SYS__exit), semafori e lock.

WAITPID

Si vuole realizzare il support per la system call `waitpid` che permette a un processo di attendere il cambio di stato di un altro processo di cui sia noto l'identificatore (pid).

Per semplicità si chiede di gestire solo il cambiamento di stato a processo terminato (tralasciare perciò eventuali resume connessi a signals).

Dopo il `tread_exit` (vedi `SYS__exit`), il processo resta in stato "zombie" fino a che un altro processo non esegue una wait/waitpid (in OS161 solo `waitpid`) e ne ottiene lo stato di uscita.

WAITPID

Il laboratorio può essere suddiviso in più parti. Si consiglia di verificare la correttezza di ogni singola parte (tramite debugger) prima di passare alla successive.

- Attesa della terminazione di un user process con ritorno di exit status
- Distruzione della struttura dati del processo
- Assegnazione di pid al processo
- (opzionale) realizzare `getpid` e `fork`

TERMINAZIONE DI UN PROCESSO

Si consiglia di realizzare inizialmente una funzione kernel `int proc_wait (struct proc *p)` che gestisca, mediante semaforo (aggiunti come campo alla struttura proc), l'attesa della fine (con chiamata alla SYS__exit) di un altro processo di cui si ha il puntatore alla relativa struttura. Guardare le note del laboratorio per maggiori dettagli.

DISTRUZIONE DELLA STRUTTURA PROCESSO

La struct proc non può essere distrutta finchè un altro processo chiami la waitpid e non riceva la segnalazione con status di uscita. Si consiglia di chiamare `proc_destroy` nella `proc_wait` dopo l'attesa su semaforo. Questo richiede anche la modifica di `sys__exit` che non deve distruggere la struttura dati ma ne segnala semplicemente la terminazione.

ASSEGNAZIONE DEL PID

Per l'attribuzione di un pid a un processo, occorre tener conto che si tratta di un intero unico (tipo pid_t), di valore compreso tra PID_MIN e PID_MAX (kern/include/limits.h), definiti in base a __PID_MIN e __PID_MAX (kern/include/kern/limits.h). Per l'attribuzione del pid e i passaggi da processo (puntatore a struct proc) a pid e viceversa, occorre realizzare una tabella.

TABELLA DEI PROCESSI E WAITPID

La fine di un processo con sys_exit non necessita, se ad aspettare è il kernel che ne ha il puntatore, della waitpid (con processo identificato da pid), ma è sufficiente la proc_wait (processo identificato da puntatore). La waitpid invece è necessaria per gestione di processi da parte di programmi user.