	Makefile				0.05	smmMon e smmd							0.25	libsmm.a						0.3	Prestazioni e protezione		0.05	Codice			0.2	Relazione		0.15	Voto base	3per=0pti, 2per=1pto, 1p=2pti	Max 2 punti		Voto Es. 2			Voto es. 1	Voto: 0.25*es1 + 0.75*es2	
Gruppo	MAK.ALL	MAK.INS	MAK.CLN	Qualita' Makefile	**Average**	SRV.REX	MON.REX	MON.AFOK	MON.AFER	MON.CHK	MON.CPT	MON.BF	**Average**	LIB.AFOK	LIB.AFER	LIB.CPT	LIB.PG	LIB.CAL	LIB.REA	**Average**	SMM.PER	SMM.PRO	**Average**	Qualita' codice	Commenti	Modularita'	**Average**	Struttura e informazioni	Stile testo e grafica	**Average**	**Media pesata**	**Bonus gruppo**	**Estensioni opzionali**	**penalizzazioni per errori nel codice**	**Voto Es. 2**	**linee codice e commenti**	**linee test SW**		**Voto Finale**	**NOTE**
Compito-Perfetto	10	10	10	10	**10.0**	10	10	10	10	10	10	10	**10.0**	10	10	10	10	10	10	**10.0**	10	10	**10.0**	10	10	10	**10.0**	10	10	**10.0**	**33.0**	**0**	**0**	**0**	**33.00**	4		33	**33.00**	
Alberto Rivelli 1067153	10	9	10	10	**9.8**	10	10	10	4	10	0	10	**7.7**	10	10	10	10	10	10	**10.0**	10	10	**10.0**	9	9	9	**9.0**	7	9	**8.0**	**29.4**	2	0	0	**31.42**	1554	0	26	**30.07**	
Davide Lo Re 1196176 [25] Edoardo Emili 1156893 [23]	7	10	10	9	**9.0**	10	8	9	10	5	10	10	**8.9**	10	10	10	10	10	9	**9.8**	10	10	**10.0**	5	9	9	**7.7**	9	8	**8.5**	**29.4**	1	1	0	**31.44**	2352	0	24	**29.58**	Il voto finale deve essere calcolato separatamente con i voti es. 1.
Andrea Di Giuseppe 1059420 [21]	9	10	10	9	**9.5**	10	9	10	10	10	10	10	**9.9**	9	10	10	10	10	6	**9.2**	10	10	**10.0**	8	8	8	**8.0**	9	7	**8.0**	**29.7**	2	0	0	**31.66**	2470	0	21	**29.00**	
Goffredo Vito 1215466 [27] Capuccini Marco 1205979 [NO]	4	6	10	6	**6.5**	10	10	6	6	0	0	0	**4.6**	10	10	10	10	10	10	**10.0**	5	10	**7.5**	7	7	6	**6.7**	7	7	**7.0**	**23.8**	0	0	0	**23.85**	2378	0	27	**24.63**	Il voto finale deve essere calcolato

compito perfetto 33

Rivelli	
Generale	Ottimo lavoro, professionale nell'architettura e nella forma (relazione, codice).
Makefile	Tutto OK, ma la soluzione di fare touch di file install per memorizzare l'avvenuta installazione non e' robusta, in quanto non si accorge se i file in smm vengono rimossi: dovrebbe verificare esistenza reale degli eseguibili in smm.
Test SRV e MON	SRV OK, robusto nei vari test, rimuove fifo anche con kill standard (SIGTERM). Lab PID stampa anche info su smmMem (non come spec). Alcune sequenze vanno in blocco, ad es. Test di compattamento memoria: alloc 8000000, alloc 1, free ptr di 8000000, alloc 9000000 ... non ritorna prompt. Alloc senza parametri da problemi, anche allo 0 (vedi vari casi MON.AFER). A volte dopo che e' andato in blocco, anche ripulendo tutto (kill pid e rm fifo), dopo start -d, lab dice che non e' started. Free funziona anche su indirizzo interno al blocco (non richiesto).
Test LIB	Tutto OK. Test per smmd -b e -d. Failed su alloc 5MB per -b, da rivedere.
Test SMM	Tutto OK.
Codice	Molto buono, leggibile, ben organizzato nel flusso e strutture dati.
Relazione	Buona la grafica e l'organizzazione dei contenuti. Qualche informazione potrebbe essere migliorata o approfondita, ad es.: - non si descrive perche' smmd fa fork all'inizio (a che serve?). - smmd chiude tutti i fd standard, ma se deve scrivere un log, cosa usa? Potrebbe fallire l'invio di msg su FIFO, quindi non rimane che scrivere qualcosa a video o in un file di log. - perche' uso di sbrk() per allocare smmMem? Da verificare Buddy System.
Estensioni opzionali	NO
SW aggiuntivo di test	NO
Penalizzazioni	NO
Fuori Spec	Utilizzo di semafori per controllo accesso FIFO, invece di lock. Nome fifo e' fifo... invece di FIFO...

Emili Lo Re	
Generale	Ottimo lavoro, professionale nell'architettura e nella forma, codice robusto. Scarso il controllo errori dopo syscall. Opzione per progetto: buddy system.
Makefile	touch src/bestfitter.c: compila anche daemon.o e daemon_core.o: perche'? Target test non funziona, non esiste test/test_1 Perche' target 'all' (e anche smmd, etc.) non sono in .PHONY? Qualita' del Makefile buona.
Test SRV e MON	Test SRV OK. SmmMon: start, stop, start, stop -> da' msg di errore. Lab <PID> stampa anche MEMORIA, non richiesto da spec. check <ptr> non stampa ptr all'ultimo byte.
Test LIB	Test smmd -d: OK, eccetto 1 errore LIB.REA.23. est smmd -b: i test FAILED sono LIB.CPT ma e' corretto.
Test SMM	OK.
Codice	Molto buono, leggibile, ben organizzato nel flusso e strutture dati, modulare, MA: daemon.c: calloc senza controllo errore. alloclist.c: malloc senza controllo errore. Send_data in lib.c: read, write senza controllo errore.
Relazione	README: E' stato inserito come extra support a python. Relazione, pag. 3, manca dir doc e pysmm. Sembrerebbe che nel protocollo di comunicazione si adotti sia il formato di msg tipo stringa ma anche (v. put e get) il formato tipo binario (invio byte non formattati). Perche' non uniformare tutto a binario? GENERALE: relazione molto buona, con le seguenti osservazioni: viene spiegato in dettaglio il protocolli dei msg, meno specifica e' invece la spiegazione sulle strutture dati per rappresentare la memoria allocata e libera.
Estensioni opzionali	Supporto a python. Smmd -h. Inserimento in smmMon dei comandi lettura/scrittura scalari e vettori. Vedi par. specifico nella relazione.
SW aggiuntivo di test	NO
Penalizzazioni	NO
Fuori Spec	NO

Di Giuseppe	
Generale	Ottimo lavoro, professionale nell'architettura e nella forma. Opzione per progetto: buddy system.
Makefile	OK, ma target EXE_SRV ricompila sempre tutti e tre i .c per come e' scritta la riga del gcc.
Test SRV e MON	OK. In alcuni casi (rari) ha dato memory corruption: [smmMon]> check 0x00000001 *** glibc detected *** smmMon: malloc(): memory corruption (fast): 0x09d41318 *** ======= Backtrace: ========= /lib/libc.so.6[0x8898ac] /lib/libc.so.6(__libc_calloc+0x8a)[0x88a89a] smmMon[0x80492cc] smmMon[0x8048ae1] /lib/libc.so.6(__libc_start_main+0xe0)[0x835390]
Test LIB	Test smmd -d: OK. Test smmd -b: un FAILED in LIB.AFER, si blocca su test LIB.REA.20.
Test SMM	OK.
Codice	smmMon.c: setjmp(): perche' viene usata? Calloc senza controllo errore. Codice buono, leggibile.
Relazione	README: Compilazione: perche' dalla cartella ../smm e non ./smm (e poi sarebbe meglio chiamarla directory, in Linux non esistono le cartelle!) L'albero delle dir riportato nella relazione non e' allineato con il compito consegnato: manca relazione.odt, in inc vengono listati file di tipo .c! Perche' non sono riportati i .o relativi a smmMon.c, smmd.c, etc.? Perche' nella struct UsedBlock e' inserito campo ptr_end mentre per FreeBlock si spiega che l'analogo ptr_real_end non serve, in quanto facilmente ricavabile? Come viene gestita la situazione in cui un stesso processo alloca nuovi blocchi e quelli con indirizzo piu' basso, quindi aumentando sempre l'indirizzo logico che viene assegnato e rischiando di saturare il valore del void *ptr_start oltre i 4GB? Forse il campo PC della struct Process non e' molto utile, da spiegare all'appello. Perche' alla ricezione di SIGUSR1 l'interrup handler legge prima di tutto un msg dalla FIFO? Pag. 12: l'ultima operazione nel cerchio del diagramma dovrebbe essere = (e non -) ? Le opreazioni di trasformazione ind. Logico-fisico sono da spiegare all'appello. Perche' a pag. 21 viene riportata define BEST_F ? Sono state implementati altri metodi (first fit, ...)? IN GENERALE: molto buona, ha centrato l'obiettivo di spiegare come e' stata affrontata l'implementazione piuttosto che ripetere i requisiti dati e gia' noti.
Estensioni opzionali	Realloc in smmMon.
SW aggiuntivo di test	NO
Penalizzazioni	NO
Fuori Spec	NO

Capuccini Goffredo Maggi	
Generale	Ci sono varie dimenticanze, forse dovute ad un controllo della qualita' del compito consegnato piuttosto lasco. In generale e' impostato bene e funzionante nella parte libreria. Opzione per progetto: swap out.
Makefile	Non compila! Assembler messages: Fatal error: can't create obj/inizio_smmMon.o: No such file or directory Nel tar e' stata rimossa la dir obj, ricreata da me! Rimossi tutti gli @ che prevenivano le print delle azioni, ricompila sempre, BAD!
Test SRV e MON	Alloc non stampa il ptr logico assegnato: come faccio free? Non potendo sapere i ptr logici, non si possono esguire i vari test CHK, CPT, BF.
Test LIB	Tutto OK, 2 errori perche' c'e' swap out.
Test SMM	Il test sulle performance delle put ci mette un tempo enorme. Vedi commento MVE in server.c, funz put e get: la put scrive sempre 16 MB, ecco perche' ci mette tanto, nella get mi sembra ci sia un test su buffer!=NULL inutile.
Codice	Perche' non si controlla errore della fcntl? Perche' sono ripetuti molte volte i blocchi di printf in server.c, quando sono sempre uguali o quasi? Vanno messi in una funzione a cui si passano i parametri da stampare. In server.c i codici dei cmd potrebbero essere rappresentati piu' elegantemente da define. Solo 3 sorgenti, potrebbero essere distruite meglio le funzioni.
Relazione	README estremamente stringata, poco utile. Relazione: variabile int avviato di smmMon.c: come fa ad accorgersi se il server termina a causa di un SIGKILL esterno? Pag. 2: qual e' la difficolta' nel rimuovere TUTTI i warning i compilazione? Politica di swap (p. 7): perche' si manda su disco il blocco esistente da piu' tempo, forse meglio quello che non viene acceduto da piu' tempo! Free e realloc operano anche su blocchi swappati. GENERALE: viene ripetuto incessantemente l'attibuto “semplice” a qualsiasi scelta implementativa: meglio essere piu' asettici in una relazione tecnica e poi il semplicemente e' sempre molto discutibile.
Estensioni opzionali	NO
SW aggiuntivo di test	NO
Penalizzazioni	NO
Fuori Spec	NO